Geometrische onzekerheidsmodellen voor correspondentieproblemen in digitale beeldverwerking

Geometric Uncertainty Models for Correspondence Problems in Digital Image Processing

Kristof Teelen

Promotoren: prof. dr. ir. W. Philips, dr. ir. P. Veelaert
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. H. Bruneel
Faculteit Ingenieurswetenschappen
Academiejaar 2010 - 2011

ISBN 978-90-8578-388-6
NUR 958
Wettelijk depot: D/2010/10.500/64

Geometrische onzekerheidsmodellen voor
correspondentieproblemen in digitale beeldverwerking

Geometric Uncertainty Models for Correspondence
Problems in Digital Image Processing

Kristof Teelen

Examencommissie
prof. dr. ir. Bjorn De Sutter (Universiteit Gent)
prof. dr. ir. Richard Kleihorst (VITO - Universiteit Gent)
dr. ir. Peter Lambert (Universiteit Gent)
dr. ir. Ben Minnaert (Hogeschool Gent)
prof. dr. ir. Wilfried Philips (Universiteit Gent - promotor)
prof. dr. ir. Rik Van de Walle (Universiteit Gent - voorzitter)
prof. dr. ir. Marc Van Droogenbroeck (Université de Liège)
dr. ir. Peter Veelaert (Hogeschool Gent - promotor)

Affiliatie
Onderzoeksgroep Image Processing and Interpretation (IPI)
Interdisciplinair Instituut voor BreedBandTechnologie (IBBT)
Department Telecommunications and Information Processing (TELIN)
Faculteit Ingenieurswetenschappen
Universiteit Gent

Onderzoeksgroep Visiesystemen
Departement Toegepaste Ingenieurswetenschappen
Hogeschool Gent - Associatie Universiteit Gent

Voor Goele

Summary

Many image processing algorithms, such as image registration, object recognition or tracking, require a reliable solution for the correspondence problem. This problem consists of a search for the corresponding 2D points in distinct digital images that indicate the same real 3D world point. It also involves the detection of the consistent geometric transformation that relates all correspondences in the images of complex scenes.

This PhD research presents a solution for correspondence problems with geometric primitives (such as points and straight lines) in image sequences of complex scenes with static and dynamic cameras. Our correspondence finding procedure takes the spatial localization uncertainty for these primitives into account. We must propagate the uncertainty all through the search process. We demand that our solution must be sufficiently robust and reliable to detect a consistent correspondence set in small sets of few and ill-localized points, possibly containing a large fraction of false candidate correspondences.

First, we give an illustration of a commonly accepted solution for correspondence problems in computer vision literature.

- *Extraction of interest points or features.* The most applied feature detectors at the moment are SIFT, SURF and Harris. Feature locations are commonly detected by searching for the local extrema of a measure, that expresses a large change in the intensity pattern of a local image patch. Most recently developed detectors focus on speed and repeatability.
- *Descriptors for the local image content.* The next step involves a description of the image appearance in a local neighborhood around the detected locations. For example, descriptors such as SIFT or SURF are based on a description of the distribution of intensity gradients. Then, the distance of each descriptor in the first image to all descriptors for the second image is computed. The most

similar descriptor is selected as a putative match, resulting in a set of feature correspondence pairs from the distinct images.

- *Spatial consistency.* Verify the spatial consistency of all correspondence pairs by a robust estimation procedure for the transformation model. The RANSAC-like approaches are the most applied procedures at the moment. The final result is a transformation that optimally describes the relation between the consistent correspondence pairs.

The resulting spatially consistent correspondence pairs are considered as a reliable solution for the correspondence problem. Note that the solution of a correspondence problem might not be unique, e.g., for repetitive patterns in the image data, especially when using an iterative procedure starting from small data sets.

The common approach still shows some drawbacks. The search process assumes the presence of a large set of corresponding features in distinct images of static scenes. Mostly, the localization uncertainty is not taken into account. Additionally, the comparison of feature descriptors and the RANSAC-like procedures can be computationally very demanding. The robust estimation procedures assume that problem-specific parameters can be estimated reliably beforehand, such as the number of false positives in a dataset. These procedures distinguish only one transformation model in the data, and cannot cope with an excessive fraction of false candidates.

Therefore, we propose a theoretical framework for the matching procedure that detects consistent consensus sets, while at the same time resolving some of the above drawbacks. We will consider the use of our procedure in several registration and matching applications. Considering small data sets implies that we must take the possibility of corrupted data sets into account. We assume that the corrupted sets can contain many false candidate correspondences, and/or that the feature points can be inaccurately localized.

The localization uncertainty is introduced by the feature detectors because of different influences in the image acquisition process, such as noise, discretization effect, varying illumination conditions for complex 3D structures, etc. The detected feature can be located a few pixels away from the expected position (after transformation). Thus, we propose not to consider an exact location, which is possibly extracted incorrectly, but rather use a region of likely feature locations.

In this work, we present a mathematical framework to model the localization uncertainty for feature points. Where other techniques use a

statistical approach to model the uncertainty, we propose a representation that is supported by the principles and the properties of uncertain geometry. We model the localization uncertainty of feature points as convex polygonal regions in the image space. Additionally, we introduce other geometric primitives, such as straight lines, into one common correspondence search process. For the lines we also model the spatial localization uncertainty in the positional parameter space.

The localization uncertainty for the geometric primitives will propagate into geometric reasoning chains at a higher level. We consider the influence of the localization uncertainty on the computation of the transformation parameters, as this is an important aspect of correspondence problems. Part of this work focuses on how to find a coherent representation of transformation uncertainty derived from localization uncertainty. The uncertainty in the parameter domain is represented by a set of linear equations or inequalities, i.e., a bounded convex polytope in the parameter space, delimited by half planes.

This representation offers some advantages: it is an accurate representation of uncertainty, and allows for a simple and efficient propagation of spatial uncertainty toward geometric concepts further along the reasoning chain. Our mathematical model is straightforward extendable toward different types of transformations. The uncertainty models can be easily incorporated in a RANSAC-like process, so that correct, but ill-localized samples are not overseen in the estimation procedure.

This PhD research shows how to incorporate uncertainty into a robust search procedure to solve correspondence problems. We demand parametric consistency for the geometric transformations of features. The consistency is expressed in function of the convex transformation uncertainty polytopes. We search for the largest set of correspondence pairs for which a non-empty intersection of associated polytopes still exists, resulting in the consistent correspondence set. This research strives for an efficient and robust search procedure which results in consistent geometric transformations between corresponding image points with a reliable consensus.

Given the small and possibly corrupted data sets, we propose an additional confidence measure, based on a contrario reasoning. Suppose the features are distributed randomly over the image (without being related by a transformation), how large can a consistent correspondence set be by accident? If our search procedure results in a consistent transformation relating a correspondence set of certain size, we can compute the probability that such a set could occur for a random distribution of

points. If that probability is sufficiently small, we accept our obtained result with sufficient confidence.

The evaluation of our matching procedure for artificial and practical real correspondence problems shows that the procedure returns a correct consistent correspondence set in over 95% of the experiments for small sets of 10-40 features contaminated with up to 400% of false positives and 40% of false negatives. Therefore, we conclude that the methods developed during this PhD research are beneficial and give useful results for typical image processing applications such as image registration and tracking of rigid objects.

This PhD research was conducted for the research group Vision Systems with Peter Veelaert at the department of Applied Engineering Sciences of the University College Ghent, in close collaboration with the group Image Processing and Interpretation (IPI) at the department Telecommunications and Information Processing (TELIN) of the Faculty of Engineering (FirW) at Ghent University. The work during this PhD led to 17 publications in total, of which 8 as first author. 7 publications (4 journal papers and 3 proceedings papers) are listed in the SCI of the ISI Web of Science database, 5 proceedings papers are enlisted in the CPCI of the ISI Web of Science database, and 5 proceeding papers were presented at international conferences after review.

Samenvatting

Het correspondentieprobleem oplossen is een essentiële stap in veel beeldverwerkingalgoritmes, zoals beeldregistratie, objectherkenning of tracking. In deze stap worden de overeenkomstige 2D beeldpunten in verschillende digitale beelden gezocht die hetzelfde reële punt in de 3D wereld afbeelden. De oplossing voor dit zoekproces vereist het vinden van een éénduidige geometrische transformatie tussen de overeenkomstige punten.

Dit doctoraatsonderzoek geeft een oplossing voor correspondentieproblemen met geometrische objecten (zoals punten en rechte lijnen) in beeldsequenties van complexe scènes met statische en dynamische camera's. Tijdens het oplossen van correspondentieproblemen houden we ook rekening met de spatiale lokalisatieonzekerheid voor deze objecten, die doorheen het gehele zoekproces gepropageerd moet worden. We eisen ook dat het zoekproces voldoende betrouwbaar is om een consistente correspondentieset te vinden in een kleine verzameling van slecht gelokaliseerde punten die mogelijk een groot aantal valse kandidaat-correspondenties bevat.

Eerst illustreren we kort de procedure voor het oplossen van correspondentieproblemen die op dit moment in de computervisieliteratuur algemeen aanvaard is.

- *Extractie van kenmerkende punten of features.* Op dit moment maken de meest gebruikte detectoren (zoals SIFT, SURF en Harris) een voorlopige selectie van interessante punten op basis van een maat gerelateerd aan de lokale distributie van de intensiteitwaarden. De recent ontwikkelde detectoren focussen op snelheid en herhaalbaarheid.
- *Descriptoren voor de lokale beeldinhoud.* Vervolgens wordt er een beschrijving van de lokale beeldinhoud rond de gedetecteerde locaties opgesteld, bijvoorbeeld op basis van een distributie van intensiteitsgradiënten zoals in de SIFT- of SURF-descriptor. Voor

elk van de descriptoren in het eerste beeld wordt de afstand tot alle descriptoren in het tweede beeld bepaald. De meest nabijgelegen descriptor wordt dan geselecteerd, wat resulteert in een verzameling potentiële correspondentieparen van *features* uit de verschillende beelden.

- *Spatiale consistentie.* De spatiale consistentie wordt voor alle correspondentieparen geverifieerd door een robuuste estimatieprocedure voor de relaterende transformatie. Op dit moment zijn de meest gebruikte methodes varianten op het RANSAC-algoritme. Uiteindelijk wordt een transformatie berekend die de relatie tussen de consistente correspondentieparen optimaal beschrijft.

De spatiaal consistente correspondentieparen worden beschouwd als een betrouwbare oplossing van het correspondentieprobleem, maar het zoekproces vertoont toch enkele nadelen. Merk op dat de oplossing voor een correspondentieprobleem niet noodzakelijk uniek is, bijvoorbeeld voor repetitieve patronen in een beeld zijn verschillende optimale resultaten mogelijk, zeker voor een iteratief zoekproces met kleine datasets.

Deze aanpak vereist de detectie van een groot aantal corresponderende *feature* in verschillende beelden van statische scènes. Meestal wordt er weinig of geen rekening gehouden met lokalisatieonzekerheid tijdens de verificatie van spatiale consistentie. Daarnaast is deze verificatie door robuuste procedures, net als het zoeken van de correspondenties tussen descriptoren, een computationeel intensief proces. De robuuste estimatieprocedures vereisen voor elk probleem vooraf een betrouwbare schatting van de parameters, zoals het aantal valse kandidaten in de dataset. Deze procedures zoeken een enkel model in de beschikbare data, waarbij het aantal valse kandidaten niet te groot mag zijn.

Daarom stellen wij in dit werk een procedure voor om uitgaande van kleine datasets toch een betrouwbare, consistente set van correspondentieparen te vinden, die enkele voordelen biedt in toepassingen zoals registratie en tracking. Het werken met kleine datasets impliceert dat we rekening moeten houden met mogelijk gecorrumpeerde sets. Een gecorrumpeerde set kan veel valse kandidaat-correspondenten bevatten, en/of de punten zelf kunnen inaccuraat gelokaliseerd zijn.

Lokalisatieonzekerheid wordt geïntroduceerd bij de detectoren onder de invloed van verschillende aspecten in het beeldacquisitieproces, zoals ruis, discretisatie-effecten, veranderlijke lichtinval op complexe 3D structuren, etc. De locatie van een gedetecteerde *feature* kan een paar

pixels van de verwachte positie (na transformatie) verwijderd zijn. Wij stellen dan ook voor om niet uit te gaan van een exacte locatie die mogelijk foutief geëxtraheerd werd, maar eerder een regio van mogelijke locaties te beschouwen.

In dit werk presenteren we een mathematisch model voor de onzekerheid over de locatie van specifieke en kenmerkende beeldpunten. Waar andere technieken een statistische benadering volgen om de onzekerheid te modelleren, gaan wij eerder uit van een representatie die steunt op principes en methodes van de onzekere meetkunde. We stellen de lokalisatieonzekerheid van *features* voor als convexe polygonale regio's in de beeldruimte. Daarnaast betrekken we andere geometrische objecten, zoals rechte lijnen, in een gemeenschappelijk zoekproces naar correspondenties. Ook voor de lijnen modelleren we de spatiale lokalisatieonzekerheid in de positionele parameterruimte.

De onzekerheid over de lokalisatie van de geometrische objecten zal propageren naar geometrische concepten op een hoger niveau. We beschouwen de invloed van de lokalisatieonzekerheid op de berekening van transformatieparameters, aangezien dit een belangrijk aspect is in het correspondentieprobleem. Een belangrijk deel van het werk spitst zich toe op een coherente voorstelling van de onzekerheid op transformaties, gebaseerd op de lokalisatieonzekerheid. De onzekerheid in het transformatieparameterdomein wordt dan ook voorgesteld door een convex polytoop in de parameterruimte, begrensd door lineaire ongelijkheden of halfvlakken.

Deze voorstellingswijze biedt enkele voordelen: het laat vooral een accurate representatie voor onzekerheid toe, en vereenvoudigt een efficiënte propagatie van de spatiale onzekerheid naar andere meetkundige concepten op een hoger niveau. Ons model is eenvoudig uitbreidbaar naar verschillende types van transformaties. De onzekerheidsmodellen kunnen ingewerkt worden in een RANSAC-achtige procedure, met als effect dat goede maar inaccuraat gelokaliseerde mosters niet genegeerd worden.

In dit doctoraatsonderzoek wordt de onzekerheid ingewerkt in een robuuste zoekprocedure voor het oplossen van correspondentieproblemen. We vereisen parametrische consistentie voor de geometrische transformaties van beeldpunten. De consistentie wordt uitgedrukt in functie van convexe polytopen die de onzekerheid op transformatieparameters voorstellen. We zoeken naar de grootste verzameling van correspondentieparen waarvoor nog een gemeenschappelijke doorsnede van geassocieerde transformatiepolytopen te vinden is, resulterend in

een consistente correspondentieset. Dit onderzoek streeft naar een efficiënte en tegelijk robuuste berekening van consistente geometrische transformaties tussen overeenkomstige beeldpunten met een betrouwbare consensus.

Aangezien we werken met kleine en mogelijk gecorrumpeerde sets stellen we een additionele betrouwbaarheidsmaat voor, gebaseerd op een *a contrario* redenering. Stel dat de kenmerkende punten willekeurig over het beeld gedistribueerd zijn zonder gerelateerd te zijn door een transformatie, hoe groot kan dan een consistente correspondentieset bij toeval zijn? Als onze procedure resulteert in een consistente relatie voor een correspondentieset met een zekere grootte, kunnen we de waarschijnlijkheid berekenen dat een dergelijke correspondentieset bij toeval gevormd zou zijn door willekeurig gedistribueerde punten. Indien deze waarschijnlijkheid klein genoeg is, is het resultaat voor ons voldoende betrouwbaar.

Een evaluatie van onze procedure voor het oplossen van gesimuleerde en praktische correspondentieproblemen toont aan dat een correcte, consistente correspondentieset wordt gevonden in meer dan 95% van de experimenten voor de beschouwde beelden met kleine sets van 10-40 punten gecontamineerd met valse positieven tot 400% en valse negatieven tot 40%. Daarom kunnen we besluiten dat de ontwikkelde methodes nuttige resultaten leveren in typische computervisietoepassingen zoals beeldregistratie en tracking.

Dit onderzoek werd uitgevoerd in de onderzoeksgroep Visiesystemen met promotor Peter Veelaert aan het departement Toegepaste Ingenieurswetenschappen van de Hogeschool Gent in een nauw samenwerkingsverband met de groep Image Processing and Interpretation van Wilfried Philips aan het departement TELIN van de faculteit Ingenieurswetenschappen van Universiteit Gent. Dit doctoraatsonderzoek heeft geleid tot 17 publicaties in totaal, waarvan 8 als eerste auteur. 7 publicaties (4 tijdschriftartikelen en 3 conferentiebijdragen) zijn opgelijst in de SCI van de ISI Web of Science databank, 5 conferentiebijdrages staan vermeld in de CPCI van de ISI Web of Science databank en 5 publicaties zijn na review verschenen in de proceedings van internationale conferenties.

Contents

Figures

Tables

Symbols and Acronyms

Acronyms

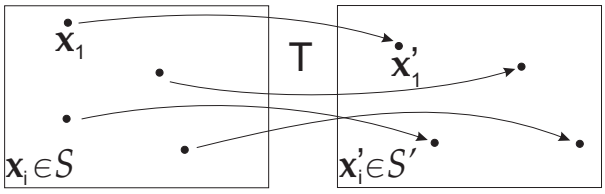
CCS	Consistent Consensus Set
CITUP	Consistent Intersection Transformation Uncertainty Polytope
CGT	Computational Gestalt Theory
CTUP	Correspondence Transformation Uncertainty Polytope
CUR	Correspondence Uncertainty Region
CVS	Consensus Verification Set
DCT	Discrete Cosine Transform
dof	degrees of freedom
DoG	Difference of Gaussians
DoH	Determinant of Hessian
DSS	Digitally Straight Segment
EBR	Edge Based Regions
FD	Feature Density
FP	False Positive
FPGA	Field-Programmable Gate Array
FN	False Negative
GCR	Global Constraint Region

GPU	Graphics Processing Unit
GTUP	Global constraint Transformation Uncertainty Polytope
IBR	Intensity-Based Regions
IMPSAC	IMPortance sampling functions using RANSAC
IUR	Implied Uncertainty Region
GLOH	Gradient Location and Orientation Histogram
LoG	Laplacian of Gaussian
LMS	Least Median of Squares
MAD	Mean Absolute Difference
MAPSAC	Maximum A Posteriori SAmple Consensus
MLESAC	Maximum Likelihood SAmple Consensus
MSAC	M-estimator SAmple Consensus
MSER	Maximally Stable Regions
PCA	Principal Component Analysis
pdf	Probability Density Function
PDE	Partial Differential Equations
PSF	Point Spread Function
PROSAC	Progressive Sample Consensus
PUP	Positional Uncertainty Pyramid
PUR	Positional Uncertainty Region
RANSAC	RANdom SAmpling Consensus
RRANSAC	Randomized RANSAC
RoI	Region of Interest
SC	Shape Context
SIFT	Scale-Invariant Feature Transform

SNR	Signal-to-Noise Ratio
SPRT	Sequential Probability Ratio Test
SURF	Speeded Up Robust Features
SSD	Sum of Squared Differences
TUP	Transformation Uncertainty Polytope

Notations

- Scalars ($a, b, \gamma, \ldots$), vectors ($\mathbf{x}, \mathbf{C}, \ldots$), and matrices ($\mathsf{T}, \mathsf{A}, \mathsf{H}$).
- A point $\mathbf{x} = [x, y]^T$ in $\mathbb{R}^2$, also used for homogeneous point coordinates, $\mathbf{x} = [x, y, w]^T$.
- A point $\mathbf{X} = [X, Y, Z]^T$ in $\mathbb{R}^3$.
- A line $\mathbf{l}$ with equation $px + qy + r = 0$ is represented by the vector $[p, q, r]^T$ of line parameters. A line $\mathbf{l}$ with equation $y - \alpha x - \beta = 0$ is represented by the vector $[\alpha, \beta]^T$ of line parameters.
- T denotes a transformation, in general, and we say that T relates the point $\mathbf{x}$ to its image (after transformation/projection) $\mathbf{x}' = [x', y']^T$ in $\mathbb{R}^2$, as $\mathbf{x}' = \mathsf{T}\mathbf{x}$.



- A correspondence or a correspondence pair is a pair of points $(\mathbf{x}, \mathbf{x}')$ that are related by a transformation T. A set of correspondences is the set of point pairs $S_c = \{(\mathbf{x_i}, \mathbf{x'_i})\}$ related by T. With two finite sets of points $\mathbf{x_i} \in S$ and $\mathbf{x'_i} \in S'$, $(\mathbf{x_i}, \mathbf{x'_i}) \in S \times S'$.

- The affine transformation $\mathbf{x}' = \mathsf{A}\mathbf{x}$:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

- The projective transformation or homography $\mathbf{x}' = \mathsf{H}\mathbf{x}$:

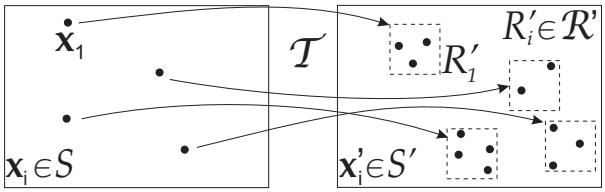
$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}.$$

- R denotes a positional uncertainty region (PUR) as a subset of the image space $\mathbb{R}^2$. R is represented as a 2D convex polygonal region bounded by n halfplanes, e.g., $r_i x' + s_i y' \geq 1, 1 \leq i \leq n$. Note that the extension of this concept in a higher dimensional space is a polytope, i.e., a bounded convex volume in an n-dimensional space enclosed by a finite number of hyperplanes. A polytope may be specified as the set of solutions to a system of linear inequalities.

- $\mathcal{T}(\mathbf{x}, R')$ denotes the transformation uncertainty set, i.e., the set of all transformations T that map a point $\mathbf{x}$ into R'. If the set can be represented as a polytope in transformation space, we denote it as transformation uncertainty polytope (TUP).

- $R'(\mathbf{x}, \mathcal{T})$ denotes the implied uncertainty region (IUR), i.e., the convex polygonal region that contains all images of $\mathbf{x}$ for each transformation in $\mathcal{T}$.

- $\mathcal{T}(S, \mathcal{R}')$ denotes the transformation uncertainty set, i.e., the set of all transformations T that map each point $\mathbf{x}$ in S into the corresponding region R' in $\mathcal{R}'$. If $\mathcal{T}$ can represented as a polytope in transformation space, we use the term transformation uncertainty polytope (TUP).



- $|C|$ denotes the cardinality of the set C, i.e., the total number of elements of the set C. For a matrix C the notation $|\mathsf{C}|$ indicates the determinant.
- An augmented matrix $[\mathsf{A}|\mathsf{B}]$ is composed by appending the rows of B to the right of the rows of A, i.e.,

$$[\mathsf{A}|\mathsf{B}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_{11} & b_{12} \\ a_{21} & a_{22} & a_{23} & b_{21} & b_{22} \\ a_{31} & a_{32} & a_{33} & b_{31} & b_{32} \end{bmatrix}.$$

- A^{+} denotes the matrix pseudo-inverse.
- The skew-symmetric matrix $[\mathbf{t}]_X$ for a vector $\mathbf{t} = (t_1, t_2, t_3)^T$ is given as

$$[\mathbf{t}]_X = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}.$$

The proofs for the several propositions in this manuscript are collected in Appendix B.

Chapter 1

Introduction

In this introductory chapter we situate the research presented in this PhD thesis. We present a novel method for modeling the positional uncertainty of geometric image features. The localization uncertainty will propagate into higher-level geometric reasoning algorithms. In this work, we mainly show how the positional inaccuracy induces uncertainty on transformations. Our methods prove their strength in correspondence problems where typically matches for various transformations must be detected in small and corrupted sets of interesting image features.

1.1 Correspondence Problems in Computer Vision

Finding correspondences is still one of the main problems in computer vision research, as many image processing applications require a solid and robust solution for matching problems. The goal of a correspondence finding or matching algorithm is to indicate for a point in one image which is the corresponding point in a second image, where both image points must show the same 3D world point.

Figure 1.1 illustrates a well-known example application: creating panoramic pictures by image stitching. This task involves a matching procedure so that subsequent images can be registered, i.e., the overlapping image parts must be placed correctly over one another.

At the moment, there exist several methods that have shown impressive results for the matching of multiple viewpoint images of static scenes. The common approaches focus on the detection and descrip-

Figure 1.1: The corresponding points in distinct images of a 3D scene.

tion of interesting points in images based on the local appearance of an image patch (e.g. [Lowe, 2004; Jurie and Schmid, 2004; Mikolajczyk et al., 2005]). The description of the appearance of the 3D scene in different images should (locally) be similar, so that corresponding points can be distinguished reliably. From the (correctly) matched pairs, one can estimate an optimal geometric transformation to register both images.

However, the matching problem is not yet solved thoroughly for more difficult problems where neither the cameras nor the scene are static. An example is the matching, recognition and tracking of moving objects in dynamic and complex outdoor scenes, which is still a difficult problem [Rothganger et al., 2007]. The robustness of the matching procedure for dynamic scenes remains a research domain which requires, and gets much attention these days.

The main part of the presented research focuses on a solid theoretical framework for several steps in a correspondence finding algorithm, to increase the robustness of the matching process for images of complex and dynamic scenes. Our matching framework should be able to cope with the uncertainty that is introduced in the imaging process by illumination variation, imperfect optics, noise, etc. Typically, the correspondences must be computed when there is little reliable information about the objects of interest, and when there are relatively high numbers of possibly false correspondences.

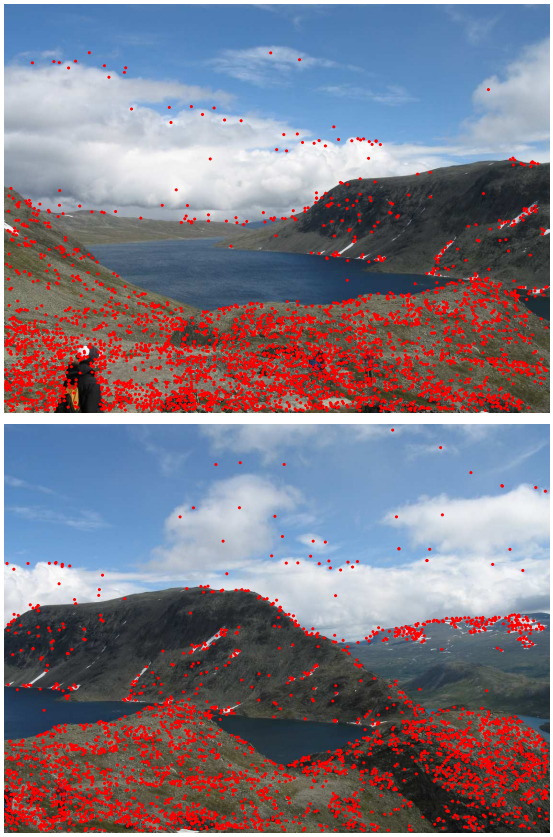


Figure 1.2: The first step of a common matching process: detecting reliable features in both images.

1.1.1 The Common Matching Approach

Currently, the common approach for solving correspondence problems for static images consists of the following steps. First a subset of characteristic points is extracted from both images by detecting remarkable patterns in the image intensity information. These points are commonly denoted as interest points or features. Figure 1.2 gives an impression of detected features for both images.

The next step involves pairing the features from one image to their counterparts in the other image. Therefore, an important property of a feature detector is its repeatability rate. Without going into detail, this means that the features corresponding to the same 3D scene point

Figure 1.3: The second step of a common matching process: finding putative matches in the second image for each feature in the first image.

should occur, and must be extracted from each image.

Most algorithms select a set of putative matches for all features in the first image, i.e., a set of possibly corresponding points in the second image, by looking at the appearance of the features. It is common practice to describe the local image intensity pattern around the features in a feature descriptor. All feature descriptors in the second image are compared to each descriptor in the first according to some similarity measure, and only (a subset of) the most alike feature(s) is selected. Figure 1.3 shows the putative best match for each feature in the first image, if there is one that is sufficiently similar.

However, the best match according to some similarity measure is not necessarily the correct match. Thus, similarity measures are not

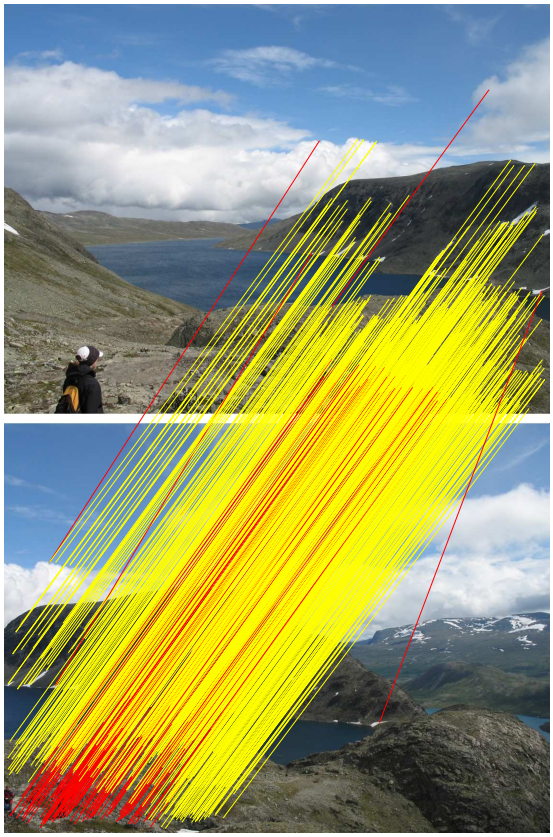


Figure 1.4: The third step of a common matching process: demanding spatial consistency. A robust model fitting algorithm iteratively determines the transformation for the given data set, and its in- and outliers (resp. yellow and red).

sufficient to solve the matching problem, and one must additionally verify the spatial consistency of the pairs of similar features. Therefore, one must verify that each relation between a corresponding pair in both images follows the rules of some geometric transformation. The data consists of inliers, data whose distribution can be explained by some set of transformation parameters, and outliers, data that do not fit. The inliers must be separated from the outliers, and the valid correspondences (related by the transformation) must be determined.

Robust model fitting algorithms can give a reliable approximation for the transformation, even though there are outliers present in the

Figure 1.5: The matching process returns a useful result for the image stitching application.

data sets. Preferably, such an algorithm can also indicate how reliable the estimate for the transformation is. Figure 1.4 shows which of the putative matches are inliers for the estimated transformation according to a robust estimation method, and which are not. The set of all correspondence pairs that are inliers for the transformation is denoted as the transformation consensus set.

Finally, the estimated transformation can be applied to register both images. One image is geometrically transformed to be aligned with the other image, as illustrated by the stitched images in Figure 1.5.

1.1.2 Matching for Complex and Dynamic Scenes

The matching and tracking of features on moving objects in dynamic and complex outdoor scenes is a difficult problem. Figure 1.6 illustrates a few exemplary situations, first for a static camera, and second, for a dynamic camera that is mounted on a moving vehicle.

For instance, consider the case of a traffic observation sequence, as illustrated in the top frames of Figure 1.6. Each car in the video will appear differently in the frames, but the features on one specific car will travel on the associated motion paths in the 3D scene. Therefore, all matches for the features on that car must be compatible with that same motion pattern. In the case where both scene and camera are static, only one consensus set must be determined, namely the correspondence set

Figure 1.6: The top images illustrate the tracking of moving vehicles in complex outdoor scenes observed by a static camera. The two next images show examples of complex outdoor scenes seen through a dynamic camera, mounted on a moving vehicle. One of the recurrent problems is the small number of reliable features on each object. Also multiple aspects of the imaging process, e.g., illumination changes, weather circumstances, camera motion, etc. will influence correspondence finding algorithms.

consistent with the transformation. If multiple objects are moving in front of a static camera, we must find a consensus set for each moving object, next to that of the background. This is difficult as there are typically few features describing each object. Also, we must be able to identify motion patterns that come to a stop (e.g. parked cars), or start moving (e.g. vehicles that drive into the imaged scene).

The correspondence problem gets increasingly difficult for a dynamic camera, i.e., when the camera is also moving, as illustrated in the bottom images of Figure 1.6. We must now identify and verify consistency for feature sets representing each moving object and each background structure in the scene. Here, perspective effects and 3D structure issues really come into play. Also, the imaging process for such dynamic scenes is influenced by parameters like illumination variation, blur, noise, etc. These parameters all have their effect and complicate the feature extraction and description procedure.

Several disturbing influences for the matching procedure are illustrated in detail in Figure 1.7. Typically, there will be few(er) features detected on each object in comparison to the total number of features detected in the whole image. For example, only 19 out of the 400 detected features in Figure 1.7(a) are located on the vehicle, which must also be distinguished from the background. Even more, as the vehicles are moving against a still background, the local image content at the vehicle outline will always be different in consecutive frames (Figure 1.7$(c-d)$). Notice that 6 of the aforementioned subset of 19 features are detected on the boundary of the car. As a solution, one could try to detect more features, either globally in the image, or locally on the vehicle. But that again raises other issues, as an even higher number of false positives, i.e., false candidate matches, is introduced in the matching process.

Figure 1.7(c) illustrates how the detection of some features is dependent on the structure of background objects, or on the shape of shadows (e.g. in the windows of the vehicle). The next frame in the sequence (d) shows that some features are located (more or less) along the motion path, but also that others are not detected, or detected elsewhere. As the motion path is not known in advance, one can look at feature descriptors to discriminate putative matches (which is a computationally costly process), and/or demand spatial consistency for a sufficiently large set of different feature pairs.

We consider to verify spatial consistency for *candidate* correspon-

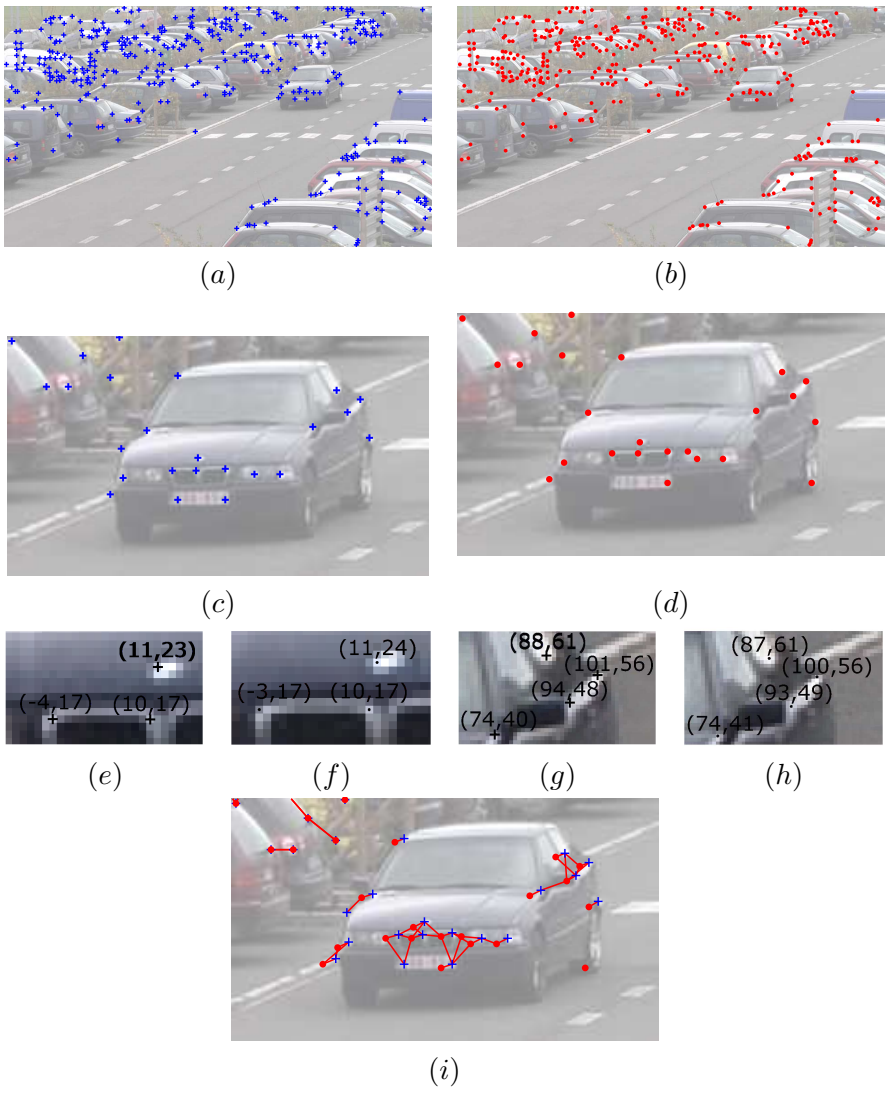


Figure 1.7: Features are detected in two consecutive frames of a video sequence $(a-b)$. $(c-d)$ There are few features detected on the car itself in both frames. $(e-h)$ Depending on the properties of the image acquisition process for this scene, the expected feature location (after transformation) can deviate a few pixels. (i) We must find the actual correspondences in a small set of possibly corrupted features, hereby considering the localization uncertainty, the false positives, etc.

dences detected in the next frame. As illustrated in Figure 1.7(i), we can consider all *nearby* features in frame(t+1) as candidate matches for a feature in frame(t). An additional problem is the uncertainty about the detected location of candidate correspondences in consecutive frames. If the feature coordinates are given in a system fixed to the vehicle (as in Figure 1.7 $(e-h)$), we notice that the extracted location can vary slightly in consecutive frames.

We can conclude that in the described tracking applications the correspondence problems must be solved when there is little reliable information about the objects of interest, and when there are relatively high numbers of possibly false correspondences. We suggest to improve the tracking by using a combined transformation model for point features and edge information. It is also important to include information regarding the 3D imaging geometry in the matching process, e.g., to put constraints on the camera parameters. We will present how to more intelligently include constraints for search regions into our correspondence finding framework, while at the same time considering an adequate model for localization uncertainty.

1.1.3 Goal

This PhD research intends to enhance several steps of a common matching process as described above. The main goal is to take the uncertainty of the detected feature locations into account in the higher-level processing steps. Due to several aspects of the imaging process, there could be a loss of accuracy and of repeatability in the feature detection process. A good *mathematical model for feature localization uncertainty* can greatly improve the performance of a correspondence or geometric reasoning algorithm. Therefore, we propagate the localization uncertainty into the parametric uncertainty of geometric transformations that map point sets onto point sets, or geometric objects onto objects.

In particular, a matching process can benefit from an uncertainty model if we a priori know additional constraints on the transformation parameters. These constraints can be translated efficiently in image *regions of interest* (RoIs) in the second image, where a match for a feature in the first image should be found. Small and accurate RoIs help to find correspondences more quickly and reliably for each feature. Computing RoIs should therefore become an essential part of real-time image and video processing.

Figure 1.8: A simple example of convex polygonal uncertainty regions for the feature locations illustrated in Figure 1.7 (d).

Additionally, we want to show how to estimate a *reliable consensus* set for a transformation from small data sets that are possibly heavily corrupted. The algorithm must be able to identify the few correct correspondences for a transformation model among a larger number of outliers. An important matter in such an algorithm is the determination of the reliability of the estimated model and its consensus set.

1.2 Our Correspondence Uncertainty Framework

Rather than working with an exact location for each feature, the positional uncertainty is modeled as a confined *uncertainty region* in which the feature is likely to occur. A simple example is given in Figure 1.8, where square uncertainty regions represent possible feature locations.

Recently, the localization uncertainty of features is mostly modeled by assuming a Gaussian distribution, which is then propagated into higher-level geometric reasoning. The uncertainty region of a feature is represented as a typical equiprobability line, i.e., an ellipse in the image space. The uncertainty on transformations is then modeled as an ellipsoid in the transformation parameter space. The most renown work in the field is done by Kanatani [Kanatani, 2005], Förstner [Förstner, 2005] and Criminisi [Criminisi, 2001].

These methods have some drawbacks. A first disadvantage are the assumptions and significant simplifications one must introduce when propagating the uncertainty. A second drawback is that an equiprobability surface often does not coincide with an ellipsoid. Our uncertainty model should accurately take the positional uncertainty of features into

account, should be easier to work with than other models (like fuzzy geometry [Bloch, 1999a,b] or statistical geometry [Kanatani, 2004; Förstner, 2005]), and should also be appropriate for image processing.

We assume that many equiprobability surfaces delimit convex regions that can be approximated well by an uncertainty polytope or polygon. The convex shape of uncertainty regions can be adequately described by a set of halfplanes. The positional representation imposes the uncertainty on the parameters of the transformation, which is represented as a polytope in the transformation parameter space. This description arises naturally when linear equations are replaced by linear inequalities in an n-dimensional parameter space.

In our geometric uncertainty approach, all uncertainty models benefit from the simplicity of a description by linear inequalities, which allows for fast and accurate computations, and a more general description of transformation uncertainty than with ellipsoids. At the same time, no numerical errors are introduced if we use multiple precision arithmetic, while it remains difficult to solve an ellipsoid intersection problem correctly.

The study of transformation polytopes and uncertainty regions originated in a larger framework of uncertain geometry that studies the properties of digitized geometric objects [Klette and Rosenfeld, 2004]. Part of the research in this domain by Peter Veelaert was aimed at extracting geometric relationships, e.g., algorithms that find and group collinear, parallel and concurrent digital line segments [Veelaert, 1999b, 2000, 2001, 2002, 2003a, 2005]. In these algorithms, geometric uncertainty modeling and its representation in both geometry and graph theory play an important role.

This PhD research extends on the main ideas of that work toward the transformation of point sets of digital objects, and the uncertainty thereof. We developed a theoretical framework to suit various matching problems, where typically correspondences must be found among a number of possible candidates in distinct feature sets. Our method should be robust, even for small and unreliable sets of features, i.e., for sets with a high percentage of outliers or false correspondences.

1.2.1 Regions of Interest

Our uncertainty framework naturally allows to develop robust algorithms for the detection and matching of geometric primitives in image

Figure 1.9: An illustration of the Region of Interest computation when the transformation parameters are known.

sequences. That is, if the position and appearance of an object in an image are more or less known, we can easily detect that object in a subsequent image provided there is a reliable model for the uncertainty of the object parameters. Thus, we must compute a region in the second image that is a good approximation of the actual location of a correspondence for the geometric in the first image. Our uncertainty model allows to induce Regions of Interest (RoIs) for object features when the transformation between both images is known, which should reduce the search space for candidate matches considerably.

A simple example is given in Figure 1.9. We can compute the transformation that relates the image of a soccer pitch to the actual 2D measures of the field. If some computer vision algorithm can determine the location of the players in the images, albeit up to some uncertainty, that location can be mapped to a top view field by that transformation. The projections of convex polygons remain polygons after transformation. From the resulting 2D view, player statistics can be derived, e.g., the distance covered during the match, or a tactical analysis. Other approaches were explored in the work by Tessens on information fusion by occupancy maps [Tessens, 2010].

As a second example, suppose that an image can be mapped on

a second image by a transformation with known parameters. This may occur for example, when two pictures of the same scene are taken from two different camera positions, for which all parameters are well-known. Suppose that the position of a line in the first image cannot be determined very precisely, but that we can define a bounded set of line parameters, describing its possible positions and angles. What are the possible line parameters in the second image? Or in other words, can we determine a RoI for the line in the second image? Once the correct correspondence is found among the possible candidates in the RoI, the transformation model can be further updated and refined.

1.2.2 Solving Correspondence Problems

When the transformation is not known, the task of a common matching framework (as illustrated in section 1.1) is to recover the transformation parameters from the image data. If the exact position of the features cannot be resolved due to errors introduced by the imaging process or by the feature detector itself, the features of one image are projected on features of a second distinct, but related image by a transformation map, that cannot be exactly determined. We will introduce a procedure that can account for slight localization errors made by a feature detector in each step of a matching procedure.

The demand for spatial consistency is commonly fulfilled by a robust model fitting algorithm such as RANSAC. RANSAC is an abbreviation for RANdom SAmple Consensus [Fischler and Bolles, 1981]. It consists of an iterative hypothesize-and-verify strategy to estimate parameters of a mathematical model from a set of observed data which contains outliers. Outliers are due to extreme values of the noise, erroneous measurements, incorrect hypotheses, etc. RANSAC tries to estimate the parameters of a model that optimally fits this data by repeatedly estimating an initial guess (the hypothesis) for the model from a (usually small) sample subset of the data. If enough inliers for an estimated model are distinguished among the other data (the verification), then the current hypothesis is a valuable candidate model to fit the observed data set.

Our matching framework incorporates the uncertainty into the consecutive steps of a feature matching strategy. We first estimate the (uncertain) position of geometric features (points, lines, circles, ...) as an allowed uncertainty region in each of both images. We explicitly take the localization uncertainty into account in the computation of a trans-

formation uncertainty polytope from a sample subset of feature pairs. Then, this uncertainty polytope is used to compute RoIs to find the actual correspondence for each feature in the first image among the candidates in the second image.

Mostly, such procedures require a lot of iterations to establish a sufficient number of correspondences, as one cannot make a reliable hypothesis (i.e. an accurately localized all-inlier sample subset) during each iteration. It is often easier and less time-consuming to compute a first estimate for the transformation based on a few prominent features in the image, albeit with some uncertainty on the transformation parameters. Also for dynamic cameras or scenes, often few features are located on each of the multiple objects. Therefore, an important aspect of our framework should be the reliable and robust computation of a transformation model even for a *small* set of feature pairs containing outliers.

Figure 1.10 illustrates our approach. We first compute a first estimate for the transformation (with uncertainty), based on a limited subset of correspondences (a). In a next step, we compute RoIs for a larger set of, perhaps less prominent, features in the image. The actual matches for each other feature are then found based on appearance in the RoIs of limited size (b). If correspondences indeed occur in the bounded RoIs, the estimate proves to be a valid transformation model. The introduction of RoIs in the matching procedure actually results in a larger set of correctly obtained correspondences than by the common approach (in Figure 1.4 up to 25% more correct correspondences are established).

Efficient and accurate algorithms must be developed for each step in the above matching procedure. Furthermore, the entire procedure should fit within a coherent matching framework. Our work is also motivated by the idea that one of the missing aspects in the common approaches is a good framework for incorporating prior knowledge on the transformation parameters of the scene and its objects. Our uncertainty framework naturally allows to include constraints on the transformation parameters in advance for applications regarding geometric feature-based registration of distinct images. These constraints will be given either in the form of uncertainty confinement regions for the feature locations or as n-dimensional polytopes for the transformation parameters.

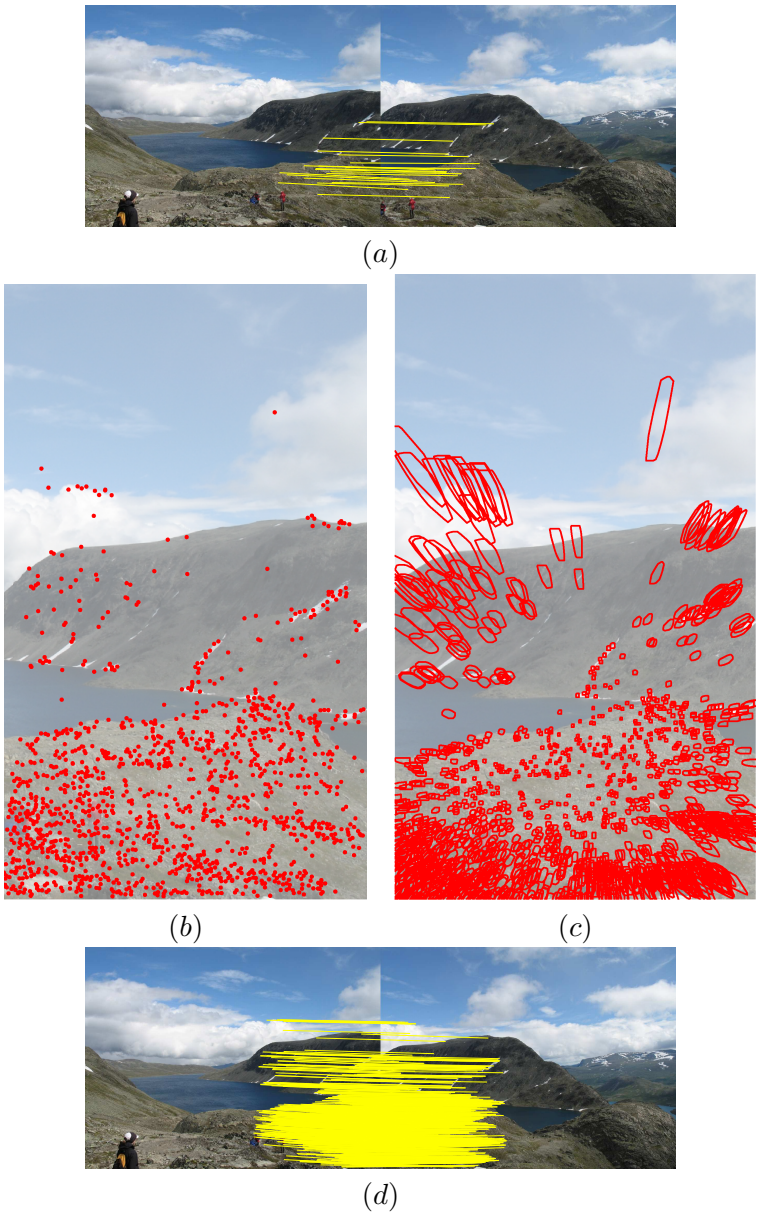


Figure 1.10: An illustration of our matching procedure. (a) First estimate a transformation uncertainty polytope from a small sample set of correspondences. $(b-c)$ Use that polytope to compute the convex polygons as RoIs for each of the detected features. (d) Determine more correspondences by appearance in the RoIs.

1.2.3 Robustness and Consistency

Our transformation uncertainty framework can prove its worth in different computer vision applications, where typically little information is available to estimate the transformation parameters, or where an initial estimate must be quickly determined from little data. Figure 1.6 shows an example which requires the tracking of moving vehicles in dynamic and complex outdoor scenes, using few features on each object. Then the motion path of different objects moving in different directions in (consecutive) images must be computed and tracked. The simultaneous tracking of different objects imposes even more stringent conditions on the robustness and computational efficiency of our method than registration problems with static images.

Techniques such as RANSAC [Fischler and Bolles, 1981] require a large set of data, and a priori assumptions, e.g., about the number of correctly occurring matches in the set. We propose a method to estimate the transformation parameters from small sets, without assumptions about the data, apart from the fact that a sufficient number of correspondences must be present in the data. Our proposed technique tries to distinguish the sets of correspondences robustly, even for small and unreliable feature sets. The most important (task-specific) parameters are taken into account by our matching procedure,

- the positional inaccuracy of a feature detector;
- the density of the feature points in the distinct images, which may include many false positives and false negatives;
- the parameters for the derivation of RoIs in which to look for candidate matches.

Eventually, when coping with such matching problems, it must be possible to assess the reliability of our procedure quickly and accurately. We use different measures to describe the reliability of our approach. The *consensus measure* for a transformation polytope is based on the number of inliers among the possible correspondences according to the transformation polytope. In itself, this is not a sufficient measure for the reliability of the correspondence sets. Not only the consensus measure should be high, the uncertainty of the returned transformation polytope should be small, and the polytope should describe a consistent set of transformation parameters for the inliers.

Our technique optimizes the model fitting process by employing parametric consistency as an additional constraint. The *consistency* of

transformations is modeled in intersection graphs, which allows for efficient computation of intersecting polytopes with consistent parameter sets. An advantage of the proposed method is that there is no need to introduce assumptions about the data beforehand. The number of inliers is estimated by the search process itself.

An additional confidence measure is based on the probability of the *random* occurrence of a correspondence set of the same size as the consensus set. If that is unlikely, the consensus set is considered as sufficiently reliable. The confidence measure can also be applied as a heuristic in the matching process, and should reduce the computational complexity.

We apply our matching procedure in several practical image processing tasks, e.g. registration and tracking as presented above. We succeed in solving such matching problems robustly, while at the same time indicating the reliability of the consensus set.

1.3 Outline

This manuscript is organized as follows. The first chapters illustrate the common state-of-the-art approach for correspondence finding algorithms in more detail. The later chapters focus on the novel ideas of this PhD, and their application in image processing tasks.

Chapter 2 considers the detection of relevant and distinguishable features. A concise overview of the different feature detection methods recently presented in literature is given, with a focus on the feature detectors of our choice for the further processing steps. This chapter also includes an illustration of how region-based processing and uncertainty are naturally inserted in the matching process.

Chapter 3 introduces a notion about what is actually meant by a correspondence, i.e., a pair of related features. Nowadays, much attention is given to the matching of adequate feature descriptors in correspondence finding algorithms, with subsequently a transformation model fit to gain geometric consistency. An initial set of possibly corresponding features is commonly extracted by comparing all feature descriptors in the second image to each descriptor in the first image. Chapter 3 starts with a short summary of a recent survey of feature descriptors. We also give an overview of robust estimation methods that fulfill the demand for spatial consistency.

Chapter 4 illustrates the different types of transformations that are used in the remainder of the manuscript: affine, homographic and perspective. This includes a short overview of their properties for different geometric primitives and their use in image processing.

Then the novel ideas of this doctoral research will be presented in more detail. First, chapter 5 considers our uncertainty model. Polygonal regions model the uncertainty on positional parameters. We show how the uncertainty on feature localization propagates into transformation uncertainty, which is represented by polytopes in the transformation parameter space. We indicate how accurate RoIs can be computed in our uncertainty framework.

In chapter 6, we show how to fit our uncertainty models into a robust estimation process, such as RANSAC. We introduce the uncertainty on transformations in the consensus and indicate how the matching process can benefit from such approach. Every correspondence problem requires computing a transformation that optimally maps the features in one image to the second. We show how to compute an optimal transformation in our framework.

Our uncertainty framework can be extended easily toward other transformation types, as shown in chapter 7 for the projective transformation.

Chapter 8 considers the demand for spatial consistency in a correspondence finding algorithm. We propose a matching tool based on parametric consistency for transformations of small and corrupted datasets using our uncertainty framework. This requires additional reliability measures for the consistent uncertainty transformation consensus sets. We present a confidence measure based on a comparison of the actual consensus set to the probability of randomly occurring consensus sets. Next, we show how to use all developed tools in selected example applications, such as registration and tracking in video sequences.

Finally, the last chapter concludes our work with some general remarks about the presented research, and an illustration of possible future work.

1.4 Contribution

The study of transformation polytopes and uncertainty regions is a continuation of the earlier work concerning the extraction of geometric relationships between digitized objects, such as lines and planes [Veelaert, 1999b, 2000, 2001, 2002, 2003a, 2005].

The PhD resulted in 17 publications in total, of which 8 as first author: 4 journal papers and 3 proceedings papers listed in the SCI of the ISI Web of Science database, 5 proceedings papers enlisted in the CPCI of the ISI Web of Science database and 5 proceedings papers in international conferences. The main contribution of the research presented in this dissertation concerns the transformation of point sets and the uncertainty thereof.

- The development of a mathematical framework for the uncertainty of transformations:

  [Teelen and Veelaert, 2005b] Computing the Uncertainty of Transformations in Digital Images, Proceedings of SPIE Vision Geometry XIII, pp. 1-12, 2005.

  [Teelen and Veelaert, 2004a] Computing the Uncertainty of Geometric Primitives and Transformations, Proceedings of ProRISC 2004, pp. 317-325.

  [Teelen and Veelaert, 2004b] Uncertainty of Affine Transformations in Digital Images, Proceedings of ACIVS 2004, pp. 23-30.

- The use of positional uncertainty in the computation of RoIs for geometric features:

  [Teelen and Veelaert, 2009] Computing Regions of Interest for Geometric Features in Digital Images, Discrete Applied Mathematics, Volume 157, Issue 16, pp.3457-3472, 2009.

  [Teelen and Veelaert, 2008] Transformation Polytopes for Line Correspondences in Digital Images, Lecture Notes in Computer Science: Combinatorial Image Analysis, Volume 4958, pp. 238-249, 2008.

- The inclusion of the transformation uncertainty framework and the confidence measures in a correspondence finding procedure for image processing applications such as image registration and tracking:

[Veelaert and Teelen, 2006a] Consensus sets for affine transformation uncertainty polytopes, Computers & Graphics, Volume 30, Issue 1, pp. 77-85, 2006.

[Teelen and Veelaert, 2005a] Confidence measures for consensus sets in transformation uncertainty, Proceedings of ProRISC 2005, pp. 679-686.

[Teelen and Veelaert, 2005c] Image Registration Using Uncertainty Transformations, Lecture Notes in Computer Science, Volume 3708, pp. 348-355, 2005.

During this PhD, a contribution was made to the research about the adaptation of difference operators to the local image content and its use for feature detection. This research topic is presented mainly in [Veelaert and Teelen, 2009a] and briefly in Appendix A. For a more elaborate discussion we refer to [Teelen and Veelaert, 2006], [Veelaert and Teelen, 2008], and [Veelaert and Teelen, 2009b].

Another contribution was made to the search for useful and elegant representations for image edge maps by geometric primitives. Both non-rigid and rigid structures can be described nicely by the fitting of first and second order geometric functions to an edge map [Veelaert and Teelen, 2006b].

This representation led to the Parabola Edge Map (PEM) a feature representation method which integrates structural and spatial information by grouping pixels of an edge map into parabola segments [Deboeverie et al., 2008]. Promising results are obtained for the tracking of PEM subsets [Deboeverie et al., 2009].

Next to geometric primitives, texture features are applied for low-level image representation. The use of Gabor-like filter banks, more specifically for the recognition of textures in scenery, led to the publications [Van Hamme et al., 2008a] and [Van Hamme et al., 2008b].

Results of the research of this doctoral thesis were used in several master theses:

- The detection and recognition of traffic signs and road marks relies on a similarity measure for consensus sets [Maertens, 2006].
- Much attention was paid to the acceleration of feature detection algorithms (Harris or SIFT) and model fitting methods, such as Radon, Hough or RANSAC, on hardware platforms like FPGA [Coppens and Van Severen, 2008] or GPU [De Gols and Pauwels, 2009].

- An interesting sideproject was the detection of moving objects in static cameras for surveillance applications [Berlamont, 2010; Haeck, 2008], soccer games [Knaepkens, 2009], and running technique analysis [Verschuere, 2010].

Two of the above master theses received the Barco-award for master theses.

Chapter 2

Features

Many image processing algorithms rely on the extraction of local image information. The extracted data must be relevant in further processing steps, hence the widely used notion of interest points or features. There exists a wide variety of feature detection procedures, each with distinct properties. As there is no consensus in recent literature about which feature detector generally performs best in which application, many methods are tailored to the desired needs of each application. As feature detection is an essential part of our matching framework, we will exploit the properties of feature detectors in correspondence finding algorithms in this chapter.

2.1 Introduction

Feature detection is a term in computer vision that refers to the detection and extraction of interesting, remarkable and relevant information for image processing applications. Instead of processing the information of all pixels, one first extracts a subset of remarkable pixels that are useful in subsequent processing steps by locally examining the image intensity information. Many computer vision tasks such as correspondence and geometric reasoning methods rely on the use of low-level features. Thus, a wide variety of detectors exists, each with specific properties, different and apt to the application of interest.

This chapter introduces the concepts and the properties of feature detection in correspondence problems, and presents some of the com-

monly applied detectors in more detail. We briefly discuss the effect of the uncertainty about the localization of the features. An essential step in this PhD is the incorporation of positional uncertainty into higher-level computations by using regions of interest, rather than just feature points.

2.2 What are Features?

Throughout literature, the term *feature* is used to indicate any location in the image which exhibits a relevant property. In the past, feature was commonly interchanged with the term *corner*. Corner was frequently used for locations which show a sufficiently high degree of variation in two dimensions (often for conventional corners such as L-corners, T- and Y-junctions). Later, it also referred to other feature types, such as black dots on white backgrounds, any location with significant 2D texture, and the intersection, join or ending of edges (where a strong variation occurs in between different image regions with a more homogeneous intensity distribution) [Schmid et al., 2000].

Nowadays, the term *feature* indicates the notion of *interest point* or *interest region*, i.e., an image location that is interesting for the application at hand. Thus, the main purpose of feature detection is to select locations (points, edges, curves or regions) in the image that are likely to be useful candidates for higher-level operations. Illustrative applications are image matching, or to describe, recognize or track objects over different images. Ideally, one would like to detect features as meaningful object parts, which is unfeasible in practice. In the earliest processing stage, one cannot yet interpret the imaged scene at such a high level. Therefore, most detectors nowadays detect characteristic local intensity patterns.

As there is no commonly accepted definition in recent literature about what a feature actually should be, its exact specification often depends on the problem at hand. However, in overview papers by Mikolajczyk, Schmid and Tuytelaars [Schmid et al., 2000; Mikolajczyk and Schmid, 2005; Tuytelaars and Mikolajczyk, 2008], there is a consensus about the properties a feature must generally comply with.

A feature is an image location that exhibits the following properties.

Localization: a well-defined and *accurately localized position* in the image. Feature detection should be a *local* operation to avoid occlu-

sion, and to allow the incorporation of simple models for geometric and illumination deformation. The localization should remain *robust* to disturbances by noise, blur, discretization, etc.

Repeatability: a sufficiently large set of features must be computed reliably in the different images with high degree of *repeatability*, i.e., is the same point extracted at precisely the same location in different images. The repeatability rate is a measure for the probability of recurrence of features in images of the observed scene. This requires positional *stability* under varying imaging conditions, including geometric (i.e. scale issues, perspective effects, ...) and illumination variations.

Invariance: the feature measures should remain *unchanged* after image transformation. A *covariant* feature changes consistently with the image transformation.

Information: the local image region around the feature must provide a *rich information content* that is sufficiently distinguishable from neighboring image regions. The image (region) must be represented efficiently by the detected features. A simple and intuitive procedure must allow to adapt the amount of extracted information.

Applicability: the use of the features must *simplify* higher-level processing. A sufficiently large number of features must be given as input for the application at hand, ideally without incorporating useless, or even false information (e.g. false positives in a matching process).

Efficiency: ideally the feature extraction process must allow for a real-time *performance* of the overall algorithm.

In the following sections, a concise overview of commonly used feature detectors is given, with most concern to repeatability and invariance. Repeatability is generally seen as one of the (or even the) most important criterion for feature detectors, certainly in algorithms for object category recognition and classification [Mikolajczyk and Schmid, 2005; Schmid et al., 2000; Tuytelaars and Mikolajczyk, 2008]. The performance of feature detectors is evaluated by the *repeatability rate*, i.e., the percentage of feature points simultaneously detected in two images compared to the overall number of detected features in an image. Repeatability explicitly evaluates the geometrical stability of feature recurrence in different images of a given scene observed under varying imaging conditions.

Ideally, features must be unaffected by geometric transformations

and changes in lighting conditions, even for moving objects in a complex non-stationary scene. Therefore, recent research concentrates on making detectors *invariant* to image transformations [Tuytelaars and Mikolajczyk, 2008]. The idea is to detect image regions covariant to a class of transformations. This can be achieved by demanding either geometric invariance, i.e., the detection methods are unaffected by geometric transformations, or robustness, i.e., make the detection methods less sensitive to small deformations, such as noise, discretization or compression effects, etc.

Of the many different feature detectors proposed in the literature, there is up to now still no consensus about which detectors are more appropriate for which image processing applications. The detection methods may vary widely in the kind of features detected, the computational complexity and the repeatability, and influence at their turn the performance of the descriptors for the detected regions of interest.

2.3 An Overview of Feature Detection Methods

An overview of the wide variety of feature detectors that exist in literature is given by Schmid [Schmid et al., 2000] and Tuytelaars [Tuytelaars and Mikolajczyk, 2008]. Most feature detection methods can be coarsely divided into three categories.

Contour based methods examine the curvature of edges in the image. A search along the connected edge chains returns meaningful locations, such as points with special characteristics, e.g., highest (change in) curvature, inflexion or intersection points, junctions, endings, etc. Interesting points can also be detected on a polygonal approximation, or on an approximation by curves or splines.

Intensity-based methods return feature locations based on a measure derived directly from the intensity values. The measures are often based on the computation of local derivatives.

Parametric model methods fit a parametric intensity model to local image regions. These methods can often provide sub-pixel accuracy, but are limited to specific types of feature models.

Next to the above categories, one can distinguish some more theoretical methods that aim at modeling the processes in the human visual system. Others extend on existing methods (or develop new methods) by explicitly using color or segmentation information. The invariance

against geometric image transformations is an important recent development.

Below, we will briefly recall some of the more interesting detection methods in the presented categories.

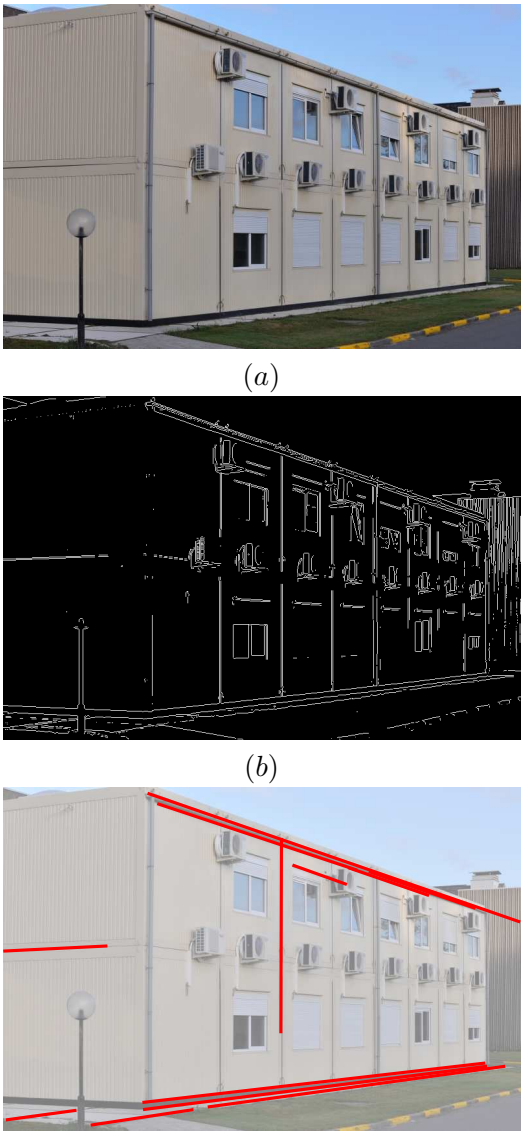
2.3.1 Contour Based Methods

Mostly, an edge detector is used as a pre-processing stage to obtain the point set for the desired contour. There is an abundance of *edge* detectors in use these days, like the Sobel, the Laplacian of Gaussian and the Canny edge detector [Canny, 1986]. A typical example of the output of an edge detector is given in Figure 2.1 (b). One can consider any location in the edge point sets where direction changes rapidly, like the intersection, junction or high curvature points as corner-like structures, i.e., features. Instead of working directly on edge information, the edges can be *parameterized*. Corners are then detected for example at points of high second derivative where the spline deviates far from the control point.

However, the assumption that corners exist along edges results in difficulties when used at junctions or intersections. Even more, it is an inadequate model when considering texture patches and point shaped features. Therefore, a large number of detectors operate directly on gray level images without requiring edge detection. These methods will be discussed thoroughly in the next section.

Not only significant point structures on the edges can be used as features, one can also directly make use of the shape extracted from the edges, e.g., geometric primitives such as straight lines (segments) or conics (circles, ellipses, parabolas). The configuration of primitives can even be a meaningful representation for objects in itself. Therefore, an often arising subproblem in image analysis is the detection of simple geometric shapes.

Several model fitting solutions exist to find the geometric shapes in the edge data. Due to imperfections in either the image data or the edge detector, there may be outliers for the desired model, such as spatial deviations from noisy edge points to the geometric model. Therefore, it is often non-trivial to group the extracted edge pixels to an appropriate set of geometric primitives. Commonly applied methods to robustly fit a geometric model (here a straight line) to the edge maps are:



(a)

(b)

(c)

Figure 2.1: (a) Original image. (b) Sobel edge image. (c) The straight lines detected by RANSAC.

Radon transform: The Radon transform in two dimensions is the integral transform, which computes the integral of a function (here the edge map) over straight lines.

Hough transform: The purpose of the Hough transform is to find imperfect instances of geometric object classes within edge points by an explicit voting procedure [Hough, 1962; Duda and Hart, 1972; Ballard, 1981]. This voting procedure is carried out in the parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space. The classical Hough transform was concerned with the identification of lines in images, and was later extended so as to also identify position, location and orientation of arbitrary shapes.

RANSAC: The hypothesize-and-verify strategy is commonly applied to detect geometric structures in edge maps [Fischler and Bolles, 1981]. Figure 2.1 (c) shows an example of the lines detected by the RANSAC method, notice that the most prominent lines in the edge image are detected.

In the field of image and shape analysis, there is a continuing and substantial interest in the problem of segmenting curves, mostly into straight line segments. Care must be taken in the extraction of line segments and their connectivity. The orientation of a segment can often be recovered quite accurately, but the segment endpoints are mostly not that reliably localized, as the topological connections between line segments are often lost during segmentation. This drawback is also visible in Figure 2.1 (c). Therefore, Rosin presented an overview and a plea for a more thorough performance evaluation for proposed and newly developed curve segmentation algorithms [Rosin, 2003, 1997].

Also in discrete geometry, curve segmentation is a subject of continuing interest. A popular approach is to segment curves into digital straight segments (DSS) [Buzer, 2005; Coeurjolly and Klette, 2002; Debled-Renesson and Reveilles, 1995; Veelaert and Teelen, 2006b]. A recurring difficulty in this segmentation approach for edge data, is the need for gapless and thinned curves, which is generally not the case. Nevertheless, there has been some effort to make detection of linear [Gao and Leung, 2002] and parabolic [Veelaert and Teelen, 2006b; Deboeverie et al., 2008, 2009] segments feasible in Canny edge maps. Figure 2.2 shows an example of the approach presented in [Veelaert and Teelen, 2006b; Deboeverie et al., 2008] to segment the Canny edge maps into straight line segments by a constructive polynomial fitting method.

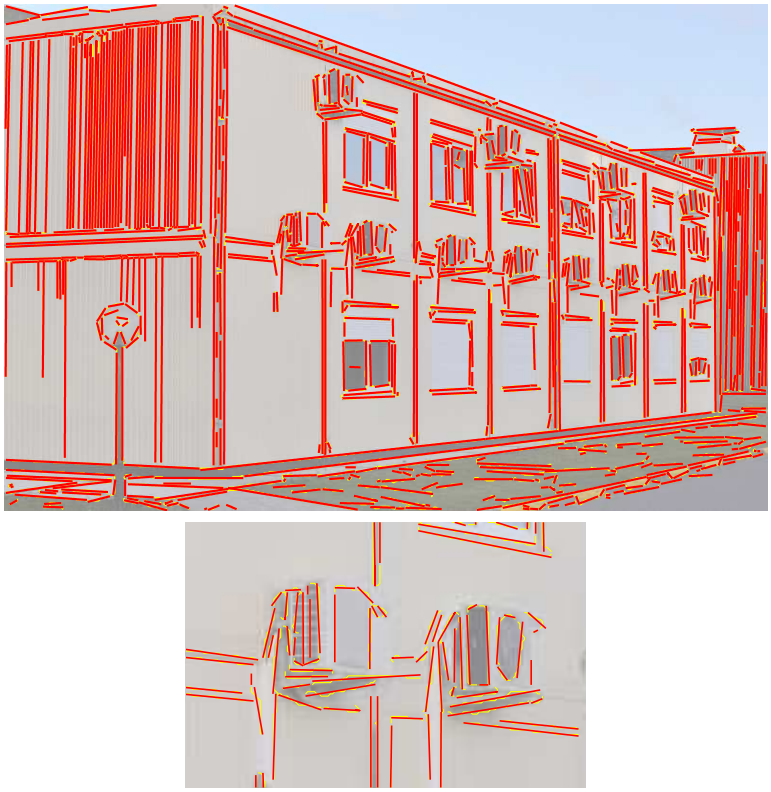


Figure 2.2: Example of the output of the constructive polynomial fitting method [Veelaert and Teelen, 2006b] for straight line segmentation from edge maps. The segments are indicated in red over the edge map in yellow.

2.3.2 Intensity-Based Methods

Features are detected by finding local extrema of a measure derived from the intensity values of the image. The measure preferably indicates image locations where there is a significant change in the intensity pattern in multiple directions, like at a corner such as in Figure 2.3. This allows to more accurately determine its unique location, and results in a richer local description of the image, than could be possible for a region of uniform intensity, or along an edge. There the local neighboring image regions are too similar in respectively any or one direction.

Mostly, the intensity surface structure of the local image patch is captured in a matrix, which is often composed of the gradient magnitude for different orientations. For example, the determinant of the Hessian (DoH) matrix is used as a measure in the methods of Beaudet

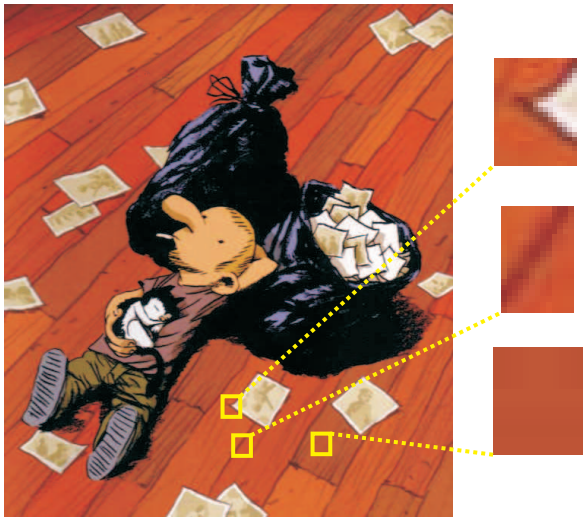


Figure 2.3: Features are preferably located where the local intensities vary in different directions, rather than at edges or in homogeneous intensity regions. Original image by Larcenet [Larcenet, 2003].

[Beaudet, 1978], Kitchen and Rosenfeld [Kitchen and Rosenfeld, 1982], SIFT [Lowe, 2004], and others. The Hessian matrix is a square matrix containing second-order derivatives of the image intensity function $I(\mathbf{x})$,

$$\mathsf{H} = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma_d) & I_{xy}(\mathbf{x}, \sigma_d) \\ I_{xy}(\mathbf{x}, \sigma_d) & I_{yy}(\mathbf{x}, \sigma_d) \end{bmatrix} \tag{2.1}$$

where the derivatives are smoothed with a Gaussian of width σ_d. The DoH operator is well adapted to detect blobs, i.e., pixel regions that are darker or brighter than their neighboring regions. Also the Laplacian of Gaussian (LoG) (which refers to the trace of the Hessian matrix) is commonly applied. Since the LoG operator also returns an elevated response on edges, the detected regions are not necessarily highly distinctive. Therefore, most recent detectors apply a cascade of operations to determine a more accurate location of discriminative features.

A large class of intensity-based methods relies on auto-correlation functions, which measures local intensity change by correlating shifted local image patches with neighboring patches. For example, in the Moravec detector [Moravec, 1977], the self-similarity of an image patch is measured by taking the Sum of Squared Differences (SSD) between an image patch and a shifted version of itself. Harris and Stephens

[Harris and Stephens, 1988] built on the approach of Moravec by approximating the second derivative of the SSD with a measure based on the second-moment matrix. Their approach avoids the use of discrete directions and discrete shifts, which is computationally more efficient and can be made isotropic.

Another major class of feature detectors works by examining small image regions to see whether their appearance mimics that of a corner. These methods check whether the intensity variations in a predefined structure verify a set of similarity rules. Different variations of similar rules are applied in different methods, such as the SUSAN [Smith and Brady, 1997] or the FAST detector [Rosten and Drummond, 2005, 2006]. The advantage of this class of methods is their computational efficiency.

Whereas the above detectors achieve translation and rotation invariance up to some extent, most recent feature detection methods such as SIFT [Lowe, 2004] and SURF [Bay et al., 2006] consider a scale space approach to obtain (at least) scale invariance. The differentials in these methods are estimated at multiple scales by the LoG, its approximation as the Difference of Gaussians (DoG), or the DoH. Other detectors explore region-based methods to obtain affine invariance, such as MSER [Matas et al., 2004] or are designed to be affine invariant (Harris-affine). However, Morel claims that none of those are fully affine invariant, and presents an affine adapted SIFT method, denoted as ASIFT, which explicitly demand invariance by simulating multiple affine views [Morel and Yu, 2009].

We will discuss some detectors of the above overview in more detail.

Harris

Harris and Stephens [Harris and Stephens, 1988] use a second-moment matrix $\mathsf{D}(\mathbf{x})$ to describe the local gradient distribution in a point neighborhood. The derivatives I_x and I_y are averaged in a square window W around a point $\mathbf{x} = (x, y)$:

$$\mathsf{D}(\mathbf{x}) = \begin{bmatrix} \sum_{\mathbf{x_k} \in W} (I_x(\mathbf{x_k}))^2 & \sum_{\mathbf{x_k} \in W} I_x(\mathbf{x_k}) I_y(\mathbf{x_k}) \\ \sum_{\mathbf{x_k} \in W} I_x(\mathbf{x_k}) I_y(\mathbf{x_k}) & \sum_{\mathbf{x_k} \in W} (I_y(\mathbf{x_k}))^2 \end{bmatrix} \tag{2.2}$$

where $I(\mathbf{x})$ is the image function and $\mathbf{x_k} = (x_k, y_k)$ are points in the window W around $\mathbf{x}$.

An improved version of the Harris detector proved to be the best detector for repeatability rate and information content, according to the

results presented in [Schmid et al., 2000]. It computes the derivatives in the matrix D more precisely than in the original method. Where originally a mask of the form $[-2\ -1\ 0\ 1\ 2]$ was used to estimate I_x, it is now replaced by derivatives computed with a Gaussian with σ_D. A Gaussian (σ_I) is used to weigh the derivatives summed over the window.

The structure of the image neighborhood is captured in $\mathsf{D}(\mathbf{x})$ as the principal changes in two orthogonal directions.

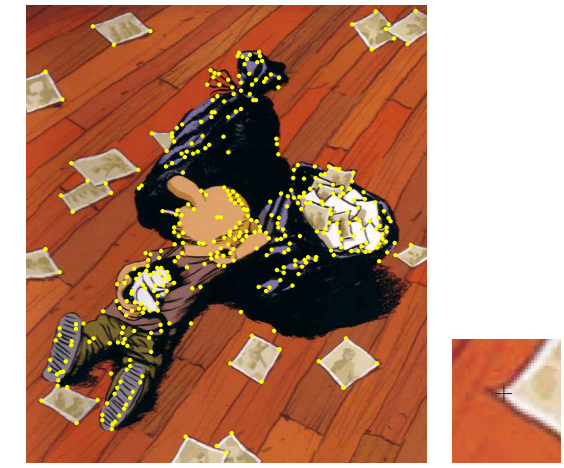
- When the intensity in the window W varies in more than one direction, both of its eigenvalues are large.
- If only one large eigenvalue occurs, an edge occurs in the neighborhood.
- The occurrence of two small eigenvalues corresponds to a homogeneous image region.

To avoid the actual eigenvalue extraction of the matrix D (which is a computationally expensive operation), the *corner strength* $C(\mathbf{x})$ in each image point $\mathbf{x}$ is measured as $C(\mathbf{x}) = \det(\mathsf{D}(\mathbf{x})) - \alpha\ \mathrm{trace}(\mathsf{D}(\mathbf{x}))^2$, where typically $\alpha = 0.06$. The second term is used to eliminate edge points with 1 strong eigenvalue.

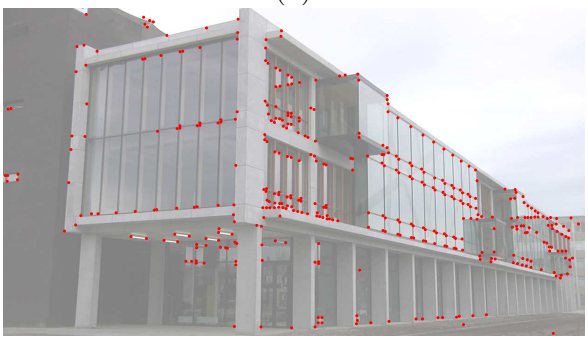
The features are selected as the points where $C(\mathbf{x})$ is above a threshold set to $x\%$ of the maximum observed corner strength. Then non-maximum suppression is applied to all $C(\mathbf{x})$. In a mask (of originally 3×3) only the maximum corner strength value is retained, while the others are discarded. This avoids the occurrence of multiple features in a small image region.

Several feature point detectors [Förstner and Gulch, 1987; Noble, 1988; Tomasi and Kanade, 1991; Shi and Tomasi, 1994; Förstner, 1994] are based on the same idea. These approaches differ in how to compute a measure for the corner strength, which have been shown to be equivalent to various matrix norms of D [Zuliani et al., 2004]. In most approaches, feature points are detected when D has two significant eigenvalues. For example, Tomasi and Kanade have shown that a good feature to track is present in an image if the eigenvalues of matrix D are both significant [Tomasi and Kanade, 1991].

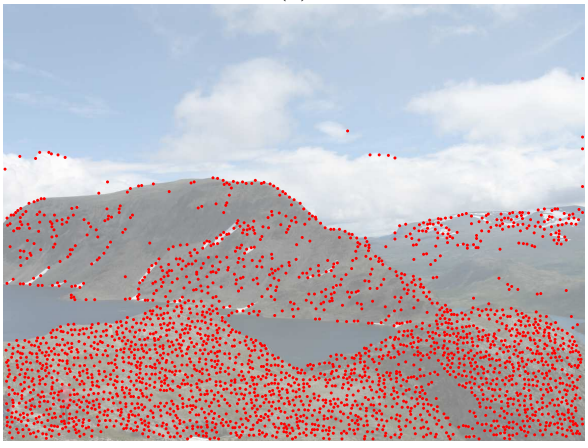
An example of the corners detected by the improved Harris detector is given in Figure 2.4. Corner features are expected to be accurately localized in such an image. Similar results are expected for a more complex static outdoor scene. We see that the detector not only responds to real corners, but also on T- or Y-junctions, and locations where there is



(a)



(b)



(c)

Figure 2.4: Example of the output of the Harris corner detector for different image types with the parameters as indicated in the text.

high curvature. Notice that Harris corner features are often extracted not exactly at the corner tip, but slightly inside the corner structure, as indicated by the enlarged example in 2.4.

In comparative studies [Schmid et al., 2000; Tuytelaars and Mikolajczyk, 2008], the Harris detector was proven to be the most repeatable and most informative among different feature detectors. Harris features are stable under varying lighting conditions, and under rotation and translation of images, and show high localization accuracy.

One of the main disadvantages of the Harris corner detector is its sensitivity to changes in image scale, so it does not provide a good basis for matching images of different sizes. However, when combined with geometric constraints or with simple descriptors, it can be used to find matches over a large image scale. Nowadays, the detector is applied in many different image processing applications, such as efficient motion tracking, camera calibration and 3D structure from motion recovery.

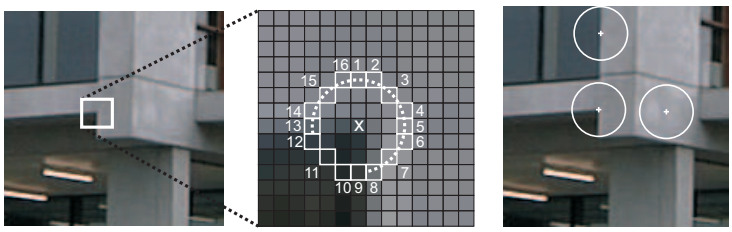
Rule-based Feature Detection

FAST. The Features from Accelerated Segment Test (FAST) detector verifies whether a pixel is a corner by looking for the appearance of one or more wedges of uniform intensity in a circular background of different intensity surrounding the candidate pixel.

On each pixel location $\mathbf{x}$ in the image a circle (e.g. of 16 pixels) is centered, as shown in Figure 2.5 (a). A feature is detected in $\mathbf{x}$ if the intensities of at least 12 contiguous pixels are all above or below the intensity of $\mathbf{x}$. One can verify whether this situation could occur by examining the points 1, 9, 5 and 13 of the circle. Candidates can already be rejected when 3 of these points do not comply with the imposed requirement. This allows for a very efficient feature detection.

To optimize the FAST detector for speed, a decision tree classifier is trained for this feature model, implemented as an if-tree structure and as such applied to the image. The FAST-ER detector [Rosten et al., 2010] is a generalization which allows the detector to be optimized for repeatability. Figure 2.6 gives an example of the detected FAST features.

SUSAN. The approach of FAST shows some similarities to the Smallest Univalue Segment Assimilating Nucleus, or SUSAN corner detector [Smith and Brady, 1997]. SUSAN starts from the concept that each image point has an associated local area of similar brightness, the USAN.



(a) (b)

Figure 2.5: (a) FAST: The points on a Bresenham circle of radius 3 are shown around a feature location. 12 contiguous pixel intensities are all above that of $\mathbf{x}$ by some threshold. (b) SUSAN: The USAN area is the area of the circle that has a similar intensity value as the circle center pixel.

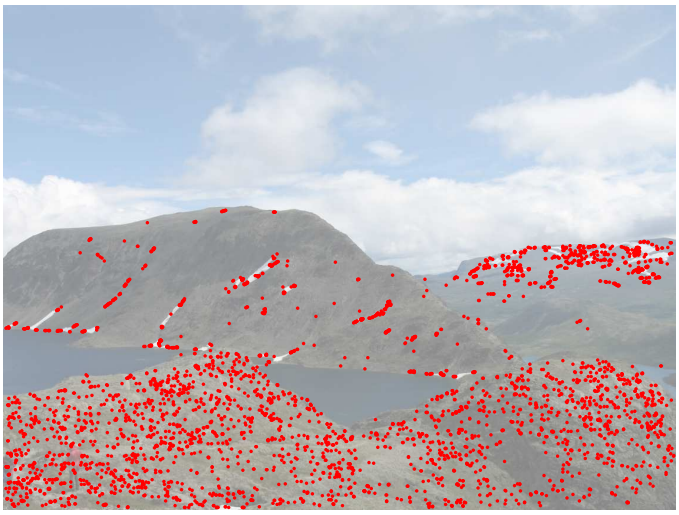


Figure 2.6: The detected FAST features.

That USAN should give a good indication of how the image structure behaves locally.

SUSAN starts from a circular mask M (of usually 37 pixels) with nucleus $\mathbf{x}$, i.e. the pixel $\mathbf{x}$ to be tested. Each pixel intensity $I(\mathbf{x_m})$ in the mask is compared to the nucleus' intensity $I(\mathbf{x})$ to find the area of the USAN. The USAN area is at a maximum when the nucleus lies in a *flat* homogeneous region of the image surface. The area falls to about half of this maximum in the vicinity of a straight edge, and decreases even further on a corner. These 3 cases are illustrated in Figure 2.5. Thus, the SUSAN operator is set to give larger response for smaller USAN areas.

Self-similarity can also be measured using a circle instead of a disc [Trajković and Hedley, 1998]. If the self-dissimilarity is small in all orientations, then the point is not classified as a corner.

Rule-based approaches for feature detection differ from other well-known methods in that no image derivatives are used, which might be sensitive to noise. This approach allows for a very efficient implementation of feature detectors.

Invariance in Feature Detection

Most input images for a vision system are subject to perspective distortions and changes in imaging conditions when observed from another location at a different time instant. Recent literature [Tuytelaars and Mikolajczyk, 2008] gives much attention to the construction of detectors invariant to intensity and geometric transformations.

A first approach to achieve *scale invariance* was detecting features in scale space, i.e., a pyramid representation obtained by subsequent low pass filtering. Mikolajczyk introduced a scale-invariant extension on the Harris detector (*Harris-Laplace*), and a scale-invariant blob-detector (*Hessian-Laplace*). The Harris-Laplace approach considers first a multi-scale detection of Harris corners, followed by automatic scale selection based on the LoG [Mikolajczyk and Schmid, 2004]. The location of a LoG-extremum proved to be particularly stable over different scales [Lindeberg, 1994], and is more robust than its single-scale counterpart. SIFT [Lowe, 2004] and SURF [Bay et al., 2006] exploit the fact that a DoG kernel provides a close approximation for the LoG, while being much faster to compute.

Affine invariant interest points can be obtained by applying affine shape adaptation. In practice, affine invariant detectors can be seen as a generalization of scale-invariant detectors for non-uniform scaling and skew. The extension of the Harris-Laplace detector by affine normalization is addressed as the *Harris-Affine* detector [Mikolajczyk and Schmid, 2002]. ASIFT explicitly covers invariance for the 6 parameters of an affine transformation by computing SIFT features in simulated sample views of the original image [Morel and Yu, 2009], which leads to more robust results.

Other approaches achieve invariance by extracting image regions. The structure-based method *Edge-Based Regions* (EBR) extracts specific parallelograms based on Harris corners and nearby intersecting edges

[Tuytelaars and Van Gool, 2004]. Both *Maximally Stable Extremal Regions* (MSER) [Matas et al., 2004] and *Intensity-Based Regions* (IBR) [Tuytelaars and Van Gool, 2000] are intensity-based methods which explore local intensity profile extrema on multiple scales. The MSER method selects image regions which are all be of a higher (or lower) intensity than surrounding pixels by growing and merging connected components in level sets. MSER allows for one of the most efficient implementations of affine invariant detectors at the moment.

Typically these methods return homogeneous regions, which is disadvantageous considering information content. However, the shape of the region boundary is usually sufficiently discriminative (if the covered image area is sufficiently large). Their approach relates to methods applied in image segmentation (such as superpixels [Ren and Malik, 2003; Ren et al., 2005]), and blob detection.

Invariance beyond affine transformations is also investigated, although projective effects are usually neglible on a local scale. A more damaging influence comes from the effect of non-planarities or non-rigid deformations. A few approaches were presented to overcome such problems, e.g., [Ling and Jacobs, 2005].

SIFT

The Scale-Invariant Feature Transform (SIFT) was proposed by David Lowe [Lowe, 2004]. The extraction of SIFT features can be seen as a cascade filtering approach, where computationally more expensive operations are applied only at locations that pass an initial test.

In a first step, the extrema of the DoG in scale space are detected. The DoG is a close approximation for the scale-normalized LoG, whose local extrema prove to result in more stable feature locations when compared to other feature measures [Lindeberg, 1994; Mikolajczyk and Schmid, 2002].

At each scale the image $I(x, y)$ is convolved $G(x, y, \sigma) * I(x, y)$ with a Gaussian,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.3}$$

Over different scales a DoG function $D(x, y, \sigma)$ is computed very efficiently as the difference of two Gaussian blurred images at different scales,

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y). \tag{2.4}$$

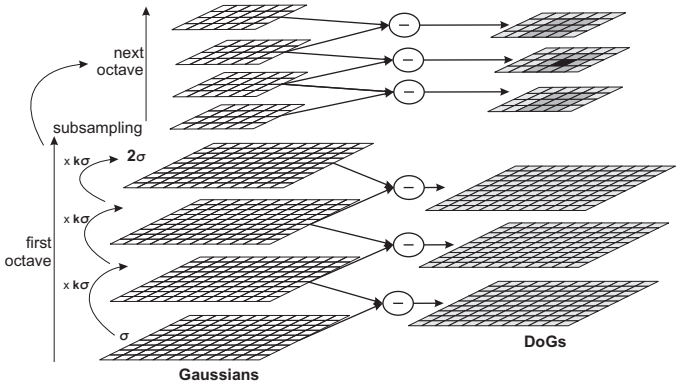


Figure 2.7: In each octave of scale space, the image is repeatedly convolved with Gaussians to produce the set of scale space images on the left. Each pair of adjacent Gaussian images is subtracted to produce the DoGs on the right. In each next octave, the initial Gaussian image is down-sampled by a factor 2. The extrema of the DoGs are detected by comparing each location to its 26 neighbors in a $3 \times 3 \times 3$ neighborhood in the current and both adjacent DoGs.

with k some constant multiplicative factor.

Figure 2.7 shows how the Gaussian blurred images are incrementally grouped per octave in scale space. To obtain an integer number s of convolved images per octave, k is selected as $k = 2^{1/s}$. The DoG images are obtained from adjacent Gaussian-blurred images in each octave. Once an octave is processed, the Gaussian image with scale 2σ is subsampled by a factor 2.

At this stage, *candidate* SIFT features are localized as the extrema in a $3 \times 3 \times 3$ neighborhood in scale space. The extrema in the DoG images identify potentially interesting locations that are invariant to both scale and orientation. Lowe's experiments show that a decrease in spacing in between consecutive DoG images (i.e. more DOG images per octave, or higher s) does not improve the repeatability rate. Many more local extrema will be detected with the increased sampling in scale space, but these are on average less stable.

A trade-off between the desired number of features and the related computational cost must be made for each application. Experiments have shown that a useful subset of stable features can already be detected from a coarse scale sampling, e.g., 3 scales per octave. Also in the spatial domain, a trade-off is required between the sampling in the image domain and the smoothing cost, which increases for larger σ.

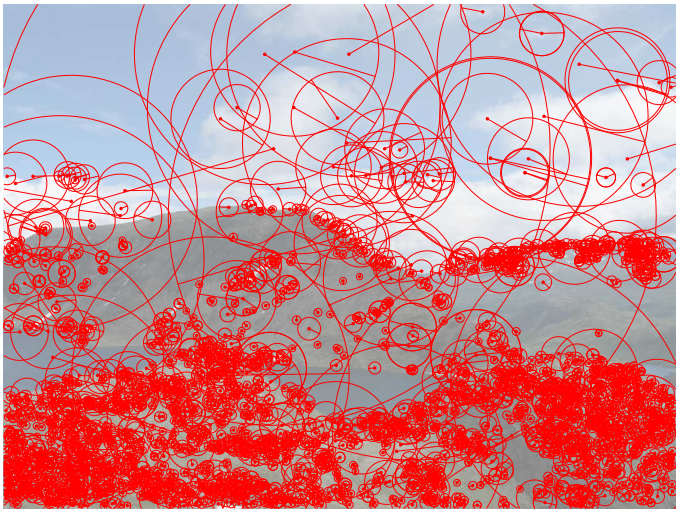


Figure 2.8: The detected SIFT features with an indication of scale and orientation as circle size and indicated axis.

The next stage of the detection process involves a more accurate and stable localization of the candidate features. At each candidate location, a detailed 3D quadratic Taylor expansion of the DoG function $D(x, y, k\sigma)$ is fitted to the local sample points to determine the interpolated location and scale of the extremum. Also all candidate extrema near edges or with a low contrast measure are discarded.

The last stage involves the assignment of an orientation parameter to each SIFT feature, next to the location (x, y) and scale $(k\sigma)$ parameters. The orientation parameter is computed from a gradient orientation histogram in the closest Gaussian smoothed image. Figure 2.8 shows an example of the detected SIFT features with scale and orientation indicated by the size of the circle and the indicated axis at the detected locations.

Experiments show that the detection of SIFT features is resistant to large amounts of noise, and the major cause of error is the initial location and scale detection. The computational complexity of such an operator is higher than that of a simpler detector such as Harris. The SIFT features are now widely used in applications such as object recognition or image stitching, because of their robustness.

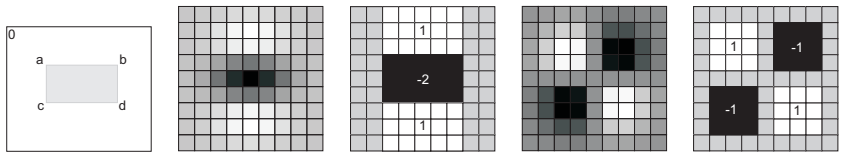


Figure 2.9: (a) The integral image approach. Each pixel $I_{\sum}(\mathbf{x})$ represents the sum of all pixels in a rectangular region defined by the image origin and $\mathbf{x}$. $I_{\sum}(\mathbf{x})$ allows to calculate the sum of all intensities in an upright rectangular region by 4 additions (here $d - c - b + a$), independent of the region size. $(b - e)$ The box filter approach of the SURF detector. The discretized and cropped Gaussian second order partial derivatives D_{yy} (b) and D_{xy} (d) are approximated by box filters (c) and (e) in a 9×9 mask. These filters can be computed efficiently by using integral images.

SURF

The Speeded Up Robust Features (SURF) approach [Bay et al., 2006] is a performant scale- and rotation-invariant feature detector and descriptor. SURF was designed to be faster than most state-of-the-art detectors, yet it approximates or even outperforms other approaches with respect to repeatability, distinctiveness, and robustness.

SURF relies on an approximation of a Hessian matrix-based measure. The determinant of the Hessian $\mathcal{H}$ in an image point $\mathbf{x}$ at scale σ is approximated as

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2. \tag{2.5}$$

Instead of computing the Gaussian second order derivatives, SURF uses box filters to obtain an approximation D_{xx}, D_{yy}, D_{xy} for the derivatives. These can be evaluated quickly with integral images [Viola and Jones, 2001], independent of the derivatives' size, and at the same time maintaining a comparable performance as with discretized Gaussians.

The main gain in performance is achieved by relying on the integral images $I_{\sum}$ in the implementation of filter operations, and also the box filter weighting coefficients are kept simple (see Figure 2.9 $(b - e)$). However, the relative weight of D_{xy} in the approximate determinant expression (Eq. 2.5) must be balanced by a factor 0.9 to correct the outcome. Also the filter response must be normalized to its mask size. Another advantage regarding computational efficiency is that SURF avoids building a scale space by iteratively smoothing by convolving

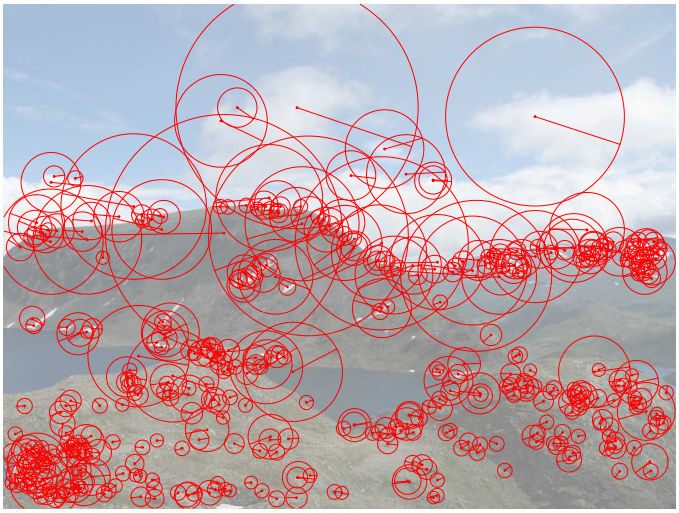


Figure 2.10: The detected SURF features with an indication of scale and orientation as circle size and indicated axis.

Gaussians. The scale space is analyzed by filtering the original image with gradually larger masks, rather than reducing the image size.

After a non-maximum suppression in a $3 \times 3 \times 3$ neighborhood, the maxima of Eq. 2.5 are interpolated in scale and image space to accurately locate the features by a method similar to that of the SIFT approach. An example of the detected SURF features is given in Figure 2.10, with an indication of the scale and orientation for each feature.

2.3.3 Evaluation

Several trends can be distinguished among the different approaches. On the one hand, a lot of effort is put into the stability of the detectors, which requires more computational effort. The accurate localization, not only to sub-pixel accuracy in image space, but also in scale space, is the subject of much attention. Also the invariance to geometric transformations is an often recurring item.

On the other hand, many approaches focus on reducing the computational complexity to gain speed for online application such as tracking. Recently, one can see a combination of both: retaining and simplifying essential parts of more complex algorithms, while still keeping a comparable performance regarding robustness and repeatability. Also, several efforts are taken to port detection algorithms to more per-

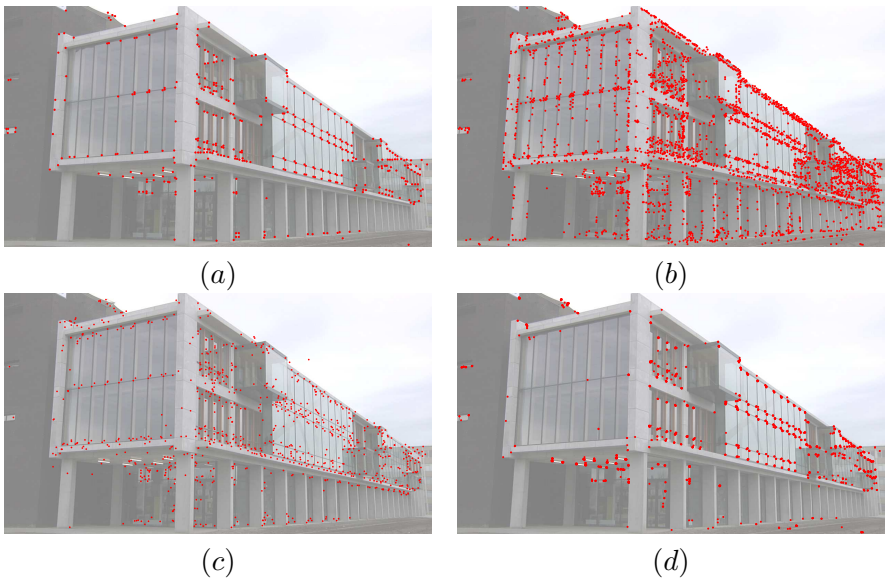


Figure 2.11: Comparison of detected feature locations for different feature detectors: Harris (a), SIFT (b), SURF (c) and FAST (d).

formant platforms, such as GPU or FPGA [Sinha et al., 2006; Se et al., 2004].

Blob detectors (like the SIFT features in 2.11 (b)) return other locations than the *corner* detectors, (such as the Harris features in Figure 2.11 (a)). Corner features are situated mostly at pronounced corner-like structures, while blob features appear in apparently more homogeneous regions surrounded by neighbors with different intensity. The location of a corner is defined as a single (exact) position in image space, while blobs are localized by their (often irregular) boundaries. Therefore, blob-like features are less accurately localized in the image space than corners, but give a better approximation for scale and shape, and are better localized in scale space. Blob detectors are said to be better suited for applications such as matching, recognition and image retrieval, while corner detectors should perform better in camera calibration or 3D reconstruction [Tuytelaars and Mikolajczyk, 2008].

In terms of scale selection, the blobs defined from scale-space extrema of the DoH (as in SIFT or SURF), have slightly better scale selection properties under non-Euclidean affine transformations than the Laplacian operator [Lindeberg, 1994]. The computational performance of the SURF detector is much better than that of the SIFT detector.

However, there is still no general consensus or generic principle by which one can detect features, no matter what the application. There are still some shortcomings in the current detector schemes. Although much desired, features are still low-level information with little semantic meaning, i.e., although features are commonly referred to as meaningful image patches, they are not yet characteristic object parts.

It is difficult to declare some feature detector as the *best* for some specific application. For example some questions one must ask are, what are typical scenes? How many features must be detected? How accurately must those features be localized?

The repeatability rate and robustness of detectors can still be improved, certainly when considering the feature detectors for a wide variety of scenes. We notice that most detectors accurately detect pronounced corner- or blob-like structures. However, a response is fired quite often in heavily textured regions, such as vegetation or foliage, or in brick walls, for which the repeatability rate is rather low.

Also effects as occlusion or background clutter come into play for many applications, when observing general 3D scenes from different viewpoints, where perspective effects can play their role. Another problem remains the detection of features on moving objects in image sequences using a static camera. Typically, a lot of information is contained in patches near the object boundary, which are perturbed by the changing background in consecutive frames. The difference in background information could influence the localization and description of the feature on the moving object. And even more problems could arise when multiple objects must be tracked in a number of non-static cameras.

In our opinion, detectors of different approaches should be combined to produce a more useful input for higher-level processing. For example in tracking applications where few corner-like structures can be detected on the moving objects, it could be advantageous to also consider line segments (e.g. on the outline of the object) or rigidly shaped regions with homogeneously distributed intensities (e.g. at the inside of objects) as features. However, the complementarity of features detectors remains an area of continued interest [Tuytelaars and Mikolajczyk, 2008]. The multiple types of feature detectors must be combined efficiently, and their outcome should be represented compactly to accurately process the increasing amount of data.

2.4 Uncertainty in Feature Detection

The application of a feature detector to an image results in a set of features that captures some local properties of the image. However, what we actually want is not the image properties, but the *3D properties* of the imaged scene. The actual appearance of the same 3D structure, e.g., the corner of a building, is imaged in 2D, and it is not unique. If the image is affected by a slight variation in one (or more) of the many properties of the scene, the image data can be substantially different. As long as there is no one-to-one relation between the image properties and the 3D scene properties, any image processing task inevitably includes some degree of uncertainty.

An extensive, but not exhaustive list of parameters influencing the imaging process includes

- the *illumination or lighting conditions*, i.e., the characteristics and location of the light source can vary over time, consider e.g., the natural daylight;
- the *camera characteristics*, e.g., the point spread function of the camera sensor, its sampling grid, quantization and noise levels;
- the *lens properties*, e.g., geometric distortion or blurring effects;
- the *camera motion*, vibration or turbulence, e.g., the quality of images obtained with a large focal length suffers from the slightest camera motion, even when mounted on a tripod;
- the *quantization* or the *discretization* effects introduced in the image acquisition process, e.g. the influence of the discretization grid;
- the *noise* introduced in the image acquisition process, i.e., mostly independent Gaussian noise is considered for image pixels;
- the *viewing orientation* and *perspective effects*;
- the *scene context* properties, e.g., the motion blur introduced by moving objects;
- the *scene structure* properties, e.g., material reflectance characteristics, and 3D shape details.

The variation in imaging condition results in the introduction of variation, or uncertainty, about the location of the detected feature. In this view, one can assess the robustness of one particular algorithm by judging its accuracy under varying imaging conditions. So in our opinion, higher-level processing steps should no longer reason in terms of exact locations, but rather of feature *uncertainty regions*.

Even a very simple experiment can show that uncertainty cannot be

avoided when considering feature detection. n features are detected by the Harris corner detector in each frame of a short video sequence of a static scene obtained with a static camera. Thus only a very limited amount of variation is expected in imaging conditions, and there is no motion in the scene. Table 2.1 presents some illustrative results for a few typical scenes.

The first 2 sequences in the table are shot outdoors with a Panasonic AG-HPX171E HD-camera on a tripod and a $13\times$ zoom lens with 28mm wide-angle setting, and the focal distance is smaller in the first than in the second sequence. Other camera parameters remained similar in both sequences. Sequence 3 and 4 were recorded indoors with a DFK 31BF03 firewire CCD color camera on a tripod, and a Computar M0814-MP 8mm F1.4 2/3" C-Mount lens. Now, the camera parameters are the same, except for the distance to the calibration pattern.

A first result is that the detector does not return the same set of n features in each frame. An analysis of the feature location in the sequence for the scene shown in Figure 2.12, shows that the 400 requested features do not always occur at the same location, but at 943 different pixels (Column **B**). This is due to 2 effects. First, as the corner strength varies slightly because of the influences on the imaging process, a different set of features may be above the threshold set to select only the n strongest features in each frame. Second, a closer look at the feature locations in each frame reveals that the detection locations form small regions where specific 3D structures are imaged.

For each sequence, we verified how many features are detected at the same pixel in 90% percent of the frames. Table 2.1 shows that this is a low percentage (column **C**), which is due to the factors affecting the imaging process. When analyzing Figure 2.12, notice that the features are not accurately localized, but their location is rather spread over a small region of neighboring pixels, typically 3×3. There are 479 connected regions of maximum size 3×3 (column **D**), thus, there are in total 479 regions in which features are located. Each feature in the region is the image of the same 3D structure.

To include only the detection of locations of local maxima in the corner strength, we look at the 3×3 regions around the n strongest features of the first frame (and not around the n strongest in each frame). We see that 93% of the detected features are located in those regions in all frames. This is given in the last column **F** of Table 2.1.

Given the relatively low numbers regarding feature location stabil-

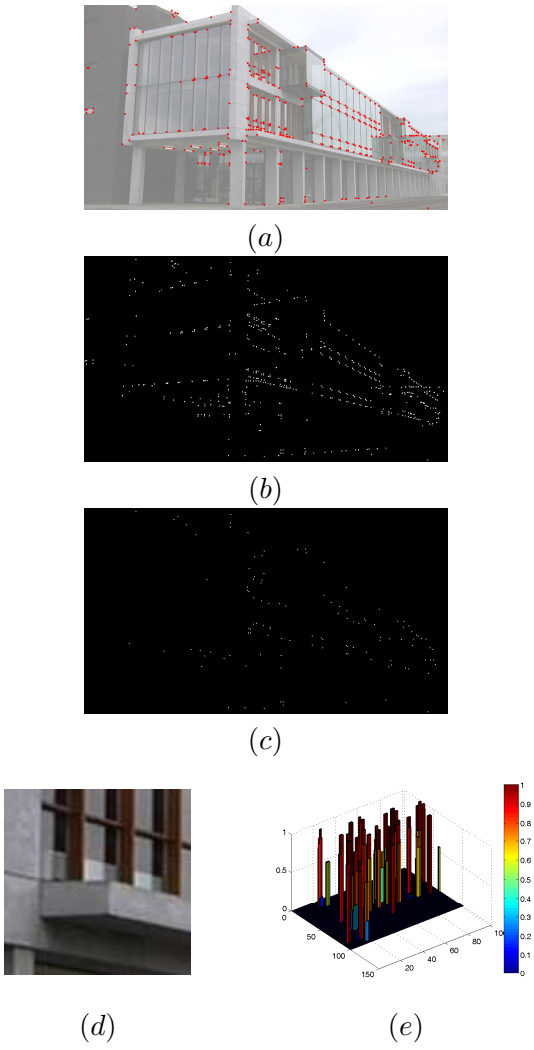
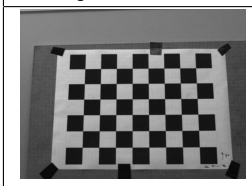
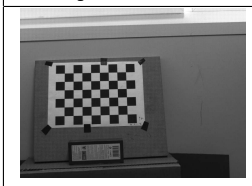


Figure 2.12: (a) The pixels where a feature is detected by the Harris corner detector in the first image of a sequence observed by a static camera in a short time span. (b) All locations at which a feature was detected in at least 1 of the sequence frames. (c) The features that are detected at the exact same pixel location in over 90% of the frames, which is only 15% of all features in (b). The detected feature locations and their relative frequency of detection in the sequence (e) for a smaller image region (d).

Table 2.1: An experiment on feature repeatability: n Harris features are computed for each frame (n is denoted in Column **A**).
B: the number of different pixel locations returned by the Harris detector, when summed over all frames in the sequence.
C: the percentage of pixel locations where a feature is detected in over 90% of the frames.
D: the number of different regions consisting of connected feature locations.
E: the percentage of regions that occur in over 90% of the frames.
F: the percentage of features detected in a 3×3 region around the location of the n strongest features in the first image of the sequence.

Dataset	**A**	**B**	**C**	**D**	**E**	**F**
 (1) 25fps,30s,960 $\times$ 540	400	943	12%	479	74%	93%
 (2) 25fps,2m,960 $\times$ 540	400	1592	9%	642	48%	79%
 (3) 15fps,1m30s,1024 $\times$ 768	100	381	2%	157	45%	69
 (4) 15fps,1m30s,1024 $\times$ 768	100	206	31%	116	81%	92%

ity in Table 2.1, even for images with such pronounced geometric content, we believe that introducing uncertainty about feature location is inevitable when reasoning or computing geometric relations from feature locations. When computing features for images from a sequence with a non-static camera, or moving objects, we expect the uncertainty measure to increase, i.e., the uncertainty regions should be enlarged, as

the accuracy of the features location will further deteriorate, due to the effects summarized above.

Considering regions instead of exact points for feature locations occurs also in other work. Tuytelaars [Tuytelaars and Mikolajczyk, 2008] points out that all features have some implicit spatial extent, as detectors must always analyze a local image neighborhood to localize features. We must not only think of a local feature as a point as defined in geometry: having a specific position in space, but no spatial extent. In practice, the smallest spatial unit in discrete images is a pixel, and discretization effects are an important influence on image acquisition and processing. We must also consider possible effects in the local feature support neighborhood.

The uncertainty of feature localization also occurs in the evaluation of feature point detectors in the work of Schmid *et al.* [Schmid et al., 2000] by the repeatability criterion. Repeated points are only taken into account for the repeatability measure if they are located in a common scene part determined by a homography H (for planar scenes). H is computed very precisely, independent of the detected features and the imaging process. Then the repeatability rate is measured by the ratio of actual matches to the total number of detected features.

The experimental results show that a repeated point is generally not detected exactly at the position $\mathbf{x}' = \mathsf{H}\mathbf{x}$, but rather in some neighborhood $R(\epsilon)$ of $\mathbf{x}'$, i.e., $R(\epsilon) = \{\mathbf{x}' | \mathrm{dist}(\mathsf{H}(\mathbf{x}), \mathbf{x}') < \epsilon\}$, with ϵ usually $1-2$ pixels. Thus, one must certainly take the localization uncertainty of feature detection into account after transformation. Even then, it is often observed that the repeatability rate is below 50%. Both the number of detected features, and the accuracy of detection vary. The performance is dependent on the application and the scene type.

Köthe [Kothe et al., 2006] has modeled the effects of the imaging process on edge segmentation. The localization error of edge pixels is explicitly derived in function of the noise level (SNR), the digitization process, and the point spread function (PSF). One of the considered methods was the detector of Canny [Canny, 1986], who presented a localization criterion in function of the estimated noise level. Recently, nonlinear filtering schemes are published to cope with the effects of noise and digitization in edge detection. Laligant [Laligant and Truchetet, 2010] described a performance criterion regarding edge pixel localization and SNR, which was evaluated in experiments on synthetic and real, noisy and degraded images.

2.5 Conclusion

Feature detection is a first low-level step of data reduction in the matching process. It results in a set of image locations with characteristic description, which are the input for higher-level processing steps, i.e., the computation of transformations. This chapter gives an overview of commonly used feature detectors in image processing. We selected several point feature detectors that appear among the best according to recent surveys and performance evaluations, for the first step in our matching framework.

Some applications retain only the exact location derived from the feature extraction process, and completely ignore the spatial extent of a feature in higher-level processing. However, region-based processing is an important issue in computer vision [Tuytelaars and Mikolajczyk, 2008], because first, region-based processing lies at the basis of almost all feature detection (and description) methods. Features are described such that they can be identified (and matched) based on the properties of a local neighborhood of pixels. And second, it arises in a natural way when describing uncertainty (also in our framework).

In our opinion, region-based processing and localization uncertainty models will gain importance in the future. Chapter 5 introduces several recently published methods, developed to cope with the localization uncertainty of features and its propagation into further computations. It inspired us to advocate the idea of modeling feature localization uncertainty as polygonal regions in the image space, as these are intuitive to use for discrete pixel neighborhoods in images, and can be used as (an accurate approximation for) a model for the actual distribution of the feature location.

Chapter 3

Correspondence of features

The main part of this work is devoted to finding a robust solution for feature matching problems. Two principal solutions are presented in literature. A first solution is to compare features by a description of local image intensity information. Many different feature descriptors have been proposed in literature. This chapter presents a short overview of commonly used descriptors in matching processes. A second method exploits the geometric spatial relations between the features in two different images to find correspondences among the two data sets. This involves the robust estimation of a transformation from the feature sets in each image. In most cases both solutions are applied together to enhance the results of the matching process.

3.1 Introduction

Most correspondence algorithms require repeatable features, and some kind of discriminative description for each feature to relate them in distinct views. Therefore, not only feature detection is an important issue in computer vision these days, feature description is also the focus of much research. The most intuitive method is to extract a local image region around the feature, and describe some well-defined properties thereof in a feature vector, or *descriptor*. The local information content is exploited by the feature descriptors to obtain a first set of matches, based on descriptor similarity. The descriptors should contain enough distinctive properties to allow for robust matching.

However, this first matching step is hardly ever sufficient to correctly distinguish all feature correspondence pairs in two distinct images. One should also demand *spatial consistency*. Therefore most algorithms require a robust model fitting algorithm to separate outliers from inliers according to some geometric transformation in the first set of matches.

3.2 Matching by Feature Descriptors

A nice overview with a performance evaluation of feature descriptors was published by Mikolajczyk and Schmid [Mikolajczyk and Schmid, 2005]. This work claims that the most important property of descriptors should be *distinctiveness*, next to *robustness* to changes in viewing conditions, and to errors of the feature detector. The distinctiveness measure is based on the likelihood of occurrence of one local descriptor within the population of all observed descriptors. The information content and the dimensionality of the descriptor are important issues in a matching procedure, just as the applied distance measure to compare descriptors.

The feature detection step could already provide the local image information for the descriptors by itself. The extraction of additional relevant information may involve quite considerable amounts of image processing. A large number of possible descriptors and associated distance measures were presented, which each emphasize different image properties like pixel intensities, color, texture, edges, etc. The many different state-of-the-art feature description techniques can be divided into four different categories of descriptors, according to the overview in [Mikolajczyk and Schmid, 2005].

Distribution-based descriptors mainly use histograms to represent local appearance characteristics. The Scale-Invariant Feature Transform (SIFT) [Lowe, 2004] and the Speeded Up Robust Features (SURF) [Bay et al., 2006] are of the best known methods.

Spatial Frequency techniques represent features in terms of the frequency content. The Fourier transform is a well-known method, but is difficult to adapt to local analysis. A better solution is offered by wavelet methods, or the Gabor transform, which, however, requires a large filter bank to capture the small frequency and orientation changes.

Local descriptors have also been evaluated in the context of texture classification by Randen and Husoy, who compare like Gabor filters, wavelet transforms, DCT, eigenfilters and others [Randen and Husoy, 1999]. Varma and Zisserman question the use of large filter banks and propose an alternative method, employing only very compact local neighborhood distributions [Varma and Zisserman, 2003].

Differential descriptors involve the properties of a local set of image derivatives (or local jet) [Koenderink and van Doorn, 1987], e.g., by convolution with Gaussian derivatives. Rotation invariance is obtained by steering the properties of the filters along the gradient orientation, like in the steerable filters approach.

Other techniques were presented, such as the generalized moment invariants for the multi-spectral description of image data by Mindru *et al.* [Mindru et al., 2004]. The generalized color moments combine local shape and color intensity distribution. There exist rational expressions of such moments that are invariant under both geometric and illumination transformations.

This extensive overview [Mikolajczyk and Schmid, 2005] remains inconclusive about which descriptors are more appropriate for which feature detectors, and how the performance of descriptors depends on the detectors. Many methods consider invariant descriptors for support regions, detected as image regions covariant to a class of transformations. Then also the question remains of which descriptor is the most appropriate to characterize support regions in a specific application?

3.2.1 Feature Descriptors

We will now consider some of the most applied descriptors in computer vision research these days in more detail.

SIFT

The *SIFT* descriptor is constructed by sampling the gradient magnitude and orientation in a (scale invariantly) detected local image neighborhood (as described in Section 2.3.2) [Lowe, 2004].

The data for each feature descriptor is collected in an array of orientation histograms, which summarizes the gradient orientations in an array of subregions of the original neighborhood region, as illustrated

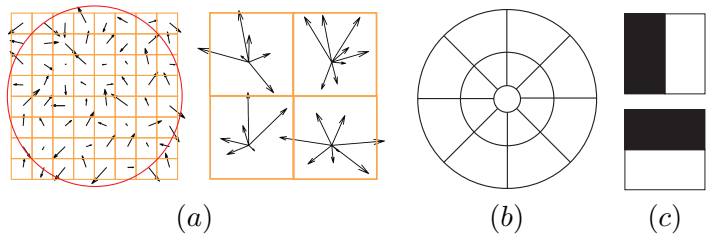


Figure 3.1: (a) A simplified example of the *SIFT* descriptor. It is created by collecting the gradient magnitude and orientation from a square sampled region centered at the feature location in an array histogram. (b) *GLOH* considers spatial regions in a log-polar location grid with 3 bins in radial direction and 8 in angular direction. (c) The Haar wavelet types used for *SURF*.

in Figure 3.1 (a). The magnitude values are weighted by a Gaussian function with $\sigma = 1.5$ times the scale of the feature, indicated by the overlaid circle (a). The sampled gradient orientations are rotated relative to the feature orientation to achieve rotation invariance. The descriptor is robust to linear and non-linear brightness changes, by adequately using pixel differences and normalization in the computations.

The dimensionality of descriptors is an important aspect to consider. The SIFT descriptor has 128 dimensions, which seems quite high, but descriptors with lower dimension perform less across a range of matching tasks, according to Lowe [Lowe, 2004]. Longer descriptors continue to perform better, but not by much. Also, they suffer from an increased sensitivity to distortion and occlusion, next to the increased computational matching cost.

SURF

SIFT uses a distribution-based descriptor of relatively low dimensionality, while still obtaining a performant discrimination rate. The SURF descriptor approach is similar to that of the SIFT descriptor, but simplified to further reduce its computational complexity.

SURF uses Haar-wavelet responses in x and y-direction in a circular neighborhood of the feature to give a reproducible orientation (Figure 3.1 (c)). The responses of the Haar wavelets can again be computed efficiently using integral images (only 6 operations are required). The responses are weighted with a Gaussian of width 2.5σ centered at the feature location on scale σ. The dominant orientation is then obtained

as the largest summated response in x and y direction in a sliding orientation window.

The descriptor is constructed from the intensity pattern in a square of height 20σ oriented along the dominant orientation. The square is divided in $n \times n$ regular smaller square subregions. The wavelet responses are computed along both sides of the subregions, and weighted by a Gaussian centered at the feature location. From the responses in each subregion, a descriptor vector is computed. The descriptor achieves invariance to an offset in illumination and contrast invariance by normalization.

One can vary the length of the descriptor (e.g. SURF-36, SURF-64 or SURF-128) by the selection of a different number n of subregions, and/or the addition of different measures based on the wavelet responses. By increasing n and the number of measures, the descriptor becomes more distinctive, and is not that much slower to compute. However, it is slower to match due to its higher dimensionality.

Others

Currently, there is quite a lot of research regarding SIFT-like methods. The PCA-SIFT descriptor [Ke and Sukthankar, 2004] captures the image gradients in a 3024 dimensional vector by sampling the gradient region at 39×39 locations in x and y direction. Then, its dimensionality is reduced to 36 with principal components analysis (PCA). This allows for faster matching procedures, but proves to be less discriminative [Mikolajczyk and Schmid, 2005].

The Gradient Location-Orientation Histogram (GLOH) extends the SIFT descriptor by changing the location grid, and using PCA in order to increase robustness and distinctiveness [Mikolajczyk and Schmid, 2005]. GLOH considers a histogram of gradient orientations in a log-polar location grid, illustrated in Figure 3.1 (b). After quantization, the gradient orientations are described by a 272 bin histogram. The higher dimensionality of the descriptor is reduced through PCA and only the n largest eigenvectors are used for description (e.g. $n = 64$ or 128).

The GLOH descriptor shows some similarities to the description used for SC [Belongie et al., 2002]. An extension of the original SC [Mikolajczyk and Schmid, 2005] uses gradient orientations for a 36 dimensional descriptor. Similar ideas are implemented as geometric histograms by Ashbrook *et al.*[Ashbrook et al., 1995].

3.2.2 Descriptor Matching

In a matching procedure, a first step is to generate putative matches for all descriptors in the first image. For each local image patch around the feature in the first image, a short list of potentially matching patches in the second image is found, based solely on its appearance.

Several common approaches for descriptor matching exist. A first option is to perform an exhaustive search, where for each feature descriptor in the first image, the distance to all features in the other image is computed. The putative matches are then the closest descriptors for an a priori selected distance measure, such as Euclidean or L_2 distance, or the Mahalanobis measure. Nearest-neighbor matching gives exact results, but is a major computational bottleneck in matching applications.

A second possible approach is a faster, but approximate nearest-neighbor search using hierarchical spatial data structures, such as KD-trees or vocabulary trees. The speed of the approximate methods comes at the cost of missing some correct matches. Moreover, their failure rate increases for large datasets.

3.2.3 Descriptor Robustness Evaluation

Mikolajczyk and Schmid [Mikolajczyk and Schmid, 2005] experimentally evaluated 10 different feature descriptors for different image transformations on different sets of images with varying content. Among the evaluated descriptors are SIFT [Lowe, 2004], SC [Belongie et al., 2002], GLOH [Mikolajczyk and Schmid, 2005], moment invariants [Tuytelaars and Van Gool, 2000], steerable filters, PCA-SIFT [Ke and Sukthankar, 2004], differential invariants [Koenderink and van Doorn, 1987], and cross-correlation.

In this experimental evaluation, the descriptors are computed for all image patches that are detected by applying the augmented and adapted Harris detector [Mikolajczyk and Schmid, 2004; Mikolajczyk et al., 2005]. This detector is invariant to scale, rotations and full affine transformations. Before computing the descriptors, all detected regions are normalized to allow for comparison, e.g., by smoothing, normalizing their orientation and compensating for illumination changes.

Recall-(1-precision) graphs are used as evaluation criterion [Mikolajczyk and Schmid, 2005]. These graphs are an adequate tool to compare

the fraction of correct matches, as indicated by the recall,

$$\text{recall} = \frac{\#\text{correct matches}}{\#\text{correspondences}}, \tag{3.1}$$

to the (relative) number of errors made in the matching procedure, which is measured by (1 - precision) as

$$(1 - \text{precision}) = \frac{\#\text{false matches}}{\#\text{correct matches} + \#\text{false matches}}. \tag{3.2}$$

For an equal dimensionality, GLOH proved to be more distinctive than SIFT in the evaluation of [Mikolajczyk and Schmid, 2005], at the cost of an increased computational complexity. GLOH obtains the best results, with SIFT a good second, for images with real geometric and photometric transformations. Both PCA-SIFT and GLOH suffer from increased complexity in the computation of the descriptor, which reduce the effect of faster matching. All three descriptors show good distinctive properties by using relative gradient-related measures from crudely localized spatial information. When the images are severely transformed, the matching results start to become less reliable. The comparison suggests a better performance of robust region-based descriptors over point-wise descriptors.

SURF was not included in that evaluation, but a similar evaluation procedure including their experiments [Bay et al., 2006] shows that the descriptors are computed much faster (up to $5\times$) than others. At the same time, the repeatability rate remains comparable or higher. SURF almost always outperforms other methods (such as SIFT, PCA-SIFT, GLOH) in these matching experiments.

3.2.4 Conclusion

From literature, we can conclude that the SIFT and SURF descriptors seem the most widely used in practical applications these days, being distinctive and rather fast at the same time. Speed can be gained by lowering the dimensionality, at the risk of losing distinctiveness. One must always take care that the effect of faster matching is not overcome by extra computations.

Then one must determine which of the putative matches are more reliable. One can introduce heuristics to define a reliability measure. As an example, one can compare the distance of the nearest neighbor to

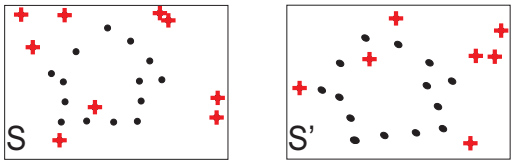


Figure 3.2: The point sets S and S' contain a number of inliers for a transformation T, i.e., point pairs for which $\mathbf{x}' = \mathsf{T}\mathbf{x}$ (dots). The outliers do not comply to T (crosses).

that of the second nearest neighbor. This ratio is expected to be higher for less distinctive features. One could e.g., retain only those features for which the reliability measure is sufficiently high.

However, the set of putative matches could still contain erroneous correspondences, possibly resulting in a high percentage of outliers. Therefore, the next step in a matching procedure consists of checking the spatial consistency for the set of putative correspondences.

3.3 Spatial Consistency in Matching Procedures

Geometric transformations play an important role in computer vision. The geometric relations between distinct static images, or between rigid objects in dynamic images, imply the constraints on the transformation of each feature.

A geometric transformation T describes the relation between corresponding points $\mathbf{x} \in S$ and $\mathbf{x}' \in S'$ in the coordinate system of 2 distinct images (or image regions). The transformation parameters must typically be estimated from a limited subset of features in each of the images. Both sets may include some correct correspondences under T, i.e., the inliers, and a number of incorrect matches according to T, i.e., the outliers. Figure 3.2 shows a simple example.

The matching process must provide a reliable estimate for T, by determining which are the correct correspondences $\{(\mathbf{x_i}, \mathbf{x_i'})\}$ between both sets, and separate those from the outliers. Typically, matching by feature descriptors is only based on similarity in appearance, and mismatches will often occur due to repetitive patterns, depth discontinuities or (self)occlusion. These outliers are sufficient to render standard least squares estimators for the transformation useless. Consequently, robust methods must be adopted which can provide a good estimate of

the solution, even if some of the data are mismatches.

Essentially, the purpose of the spatial verification step in a matching process is the robust estimation of the parameters of the image transformation model from a data set of matches, despite relatively large numbers of outliers. Several popular robust estimation techniques exist, of which RANSAC is the most used in image processing these days. We will give a detailed illustration of RANSAC and related approaches.

3.3.1 Robust Model Fitting by RANSAC

RANSAC is a very popular method in image processing, thanks to its simplicity and generality [Hartley and Zisserman, 2003]. It is a general method that is applicable to a wide range of model fitting problems, and often works well in practical applications. Therefore, we start by recapitulating the RANSAC algorithm [Fischler and Bolles, 1981].

In short, the RANSAC strategy consists of a *hypothesize-and-verify procedure*. The following steps are repeated during each loop over the set of putative matches in order to find a transformation model that best fits the data.

1. Build a hypothesis T_s for the transformation from a randomly selected sample set P_s of s putative matches. P_s must contain (at least) a minimum number of samples to be able to instantiate the model T_s that optimally explains the data in P_s.
2. Use T_s to transform all features in S, hereby predicting the location $\hat{\mathbf{x}} = \mathsf{T}_s(\mathbf{x})$ of features in the second image.
3. Verify the model by the number of inliers for T_s. The set of features that are located sufficiently close to the predicted locations $\hat{\mathbf{x}}$, is considered as the inlier set, i.e., a distance metric (e.g. Euclidean distance) lies within a specified threshold t_d for the inliers. This set is denoted as the *consensus set* C for T_s.
4. If the size of the consensus set $|C| < t_n$, repeat from 1 with another sample set.
5. Otherwise, a sufficient number of inliers is found. Refine the model T_s for all inliers, e.g., by refitting a least-squares estimate, and terminate.
6. Finally, after N hypotheses, the largest consensus set is selected, and the model is re-estimated for all its elements.

However, some disadvantages remain. First, one cannot always accurately initialize the transformation model based on the minimum

number of samples. It is possible that one of two main assumptions made in the RANSAC process [Chum, 2005] is violated. The first assumption is that an all-inlier sample should generate a hypothesis consistent with all inliers. Second, a hypothesis based on a sample set containing (at least one) outliers, should generate small support (so that correct and incorrect hypotheses can be discriminated based on the size of the consensus set). In case of a violation of these assumptions, the RANSAC procedure will run longer, or will return an incorrect solution (in the worst case). It can be caused by the distribution of the samples over the images, the ill-localization due to noise, etc.

Often, a more useful hypothesis is obtained when following a few simple rules in the selection of a sample set P_s. Select the putative correspondences such that all features are well spread over the image in a non-degenerate configuration. If possible, use prior knowledge about the transformation properties to introduce some heuristics in the search process. Even more, it is often easier and less time-consuming to select, if possible, a few of the most prominent features in the first image, for which a (more or less) reliable match can be obtained. If possible, also take the localization uncertainty into account. A careful selection of the sample set can (greatly) reduce the number of iterations during the RANSAC procedure.

Another drawback is that many iterations are required to obtain a good solution. The algorithm can only handle a moderate percentage of outliers without its computational cost escalating, while many real problems have high numbers of outliers. In these cases, a selective choice of random subsets can sometimes help.

The main shortcoming of RANSAC-like methods is the specification of the consensus set for specific problems. Much effort has been dedicated to this problem in recent literature, as shown in the next section. In this prospect, RANSAC requires relatively many parameters that must be tuned in advance to specify the probabilistic properties for the consensus set.

- The choice of the distance threshold t_d is influenced by assumptions about the noise level. If one assumes that the measurement error is Gaussian with zero mean and standard deviation σ, then one can compute t_d such that the inlier probability for the estimated model is above $x\%$. In practice, t_d is mostly chosen empirically. If the threshold is set too high, then the robust estimation can be poor.

- The fraction of inliers should be matched by the consensus set size. If the number of inliers believed to be in data is approached by the size of the consensus set, the process can be terminated. Therefore, one must be able to relate the threshold t_n to the expected inlier fraction in advance.

- The number of samples N must be chosen such that at least one out of N random selections is free from outliers with probability p (usually $p = 0.99$). If the fraction of outliers is given by e, then N samples of s features must be selected as

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}. \tag{3.3}$$

  The downside of this approach is that one must have knowledge about the expected fraction of outliers in the set, in order to compute a sufficiently reliable consensus set. N is related to the proportion, rather than the actual number of outliers, meaning that N may be smaller than the number of outliers.

  The number of samples N increases with the size of the minimal subset for a fixed probability p and a fixed probability e that a point is an outlier. One could consider to select a larger sample set P than required, because then one could obtain a better estimate, and the measuring support would more accurately reflect the true model. However, this possible advantage in measuring support is generally outweighed by the severe increase in computational cost incurred by the increase in the number of samples.

  Generally, the time complexity of RANSAC depends mostly on the complexity of the geometric model, and the number of required samples N (which is determined from the estimated fraction of outliers e, the sample set size s and the probability p).

For applications such as image registration and stitching, RANSAC is a reliable method. The putative matches are abundant, and the number of inliers should be large in comparison to the number of outliers, if the descriptor matching is sufficiently reliable. For smaller sets of features, e.g., on moving objects in tracking applications, for which the set of putative matches could be severely corrupted, it is more difficult to choose appropriate parameters in advance.

3.3.2 Other Robust Estimation Algorithms

The *Least Median of Squares* (LMS or LMedS) estimation process is another prominent approach [Rousseeuw, 1984; Zhang et al., 1995]. LMS instantiates an estimation of the transformation model by solving a nonlinear minimization problem. First, a minimum size sample set is randomly selected from the data. Where in RANSAC a model is ranked by the number of data points within a threshold distance, in LMS the median of the squared residuals of all points in the data is used. The model with least median is selected.

LMS requires no threshold setting or prior knowledge of the variance of the error, like in RANSAC. The major disadvantage of LMS is that it reaches its breakdown point when more than half of the data points are outliers, because then also the median is an outlier. One could adapt by using a different proportion to determine the selection distance, like the quartile. Thus, LMS is able to cope with a reasonably large fraction of outliers, comparable to RANSAC, but its efficiency is poor in the presence of Gaussian noise [Hartley and Zisserman, 2003].

Recently, some approaches were presented with the purpose of increasing the efficiency of a standard RANSAC procedure. These either seek to optimize the model verification process, or intervene in the sampling process, trying to generate more useful hypotheses. In the remainder of this section we will give a short overview of the most prominent methods. We notice that the inaccuracy of feature localization is rarely propagated, until recently.

In a standard RANSAC process the inliers add nothing to the score for each sample set, and for each outlier a constant penalty is added to the score. The outliers are also given a fixed penalty in *M-estimator SAmple Consensus* (MSAC) [Torr and Zisserman, 1997] (e.g. t_d^2), but the inliers are scored on how well they fit to the data hypotheses by using a simple redescending M-estimator. The sample with the lowest score is then selected as the best estimate.

MSAC is further improved in *Maximum Likelihood sample consensus* (MLESAC) [Torr and Zisserman, 2000]. MLESAC not only gives an initial estimate of the relation, but also indicates the likelihood that each correspondence is consistent with the relation. Then not only the variance on the point location must be modeled, but also the outlier distribution and prior probabilities must be estimated. The best solution is the sample for which the negative log likelihood is minimal, rather than just counting the number of inliers. The output of MLESAC can

then be used to improve the estimate of the model by a constrained optimization process. Where MLESAC was purely a maximum likelihood formulation, Maximum A Posteriori SAmple Consensus (MAPSAC) [Torr, 2002] is a more generic method in which random samples are used to maximize the (approximation of the) posterior, followed by gradient descent.

IMPortance sampling functions using RANSAC (IMPSAC) [Torr and Davidson, 2003] is also a Bayesian solution with a coarse to fine approach that appears to be beneficial in matching problems. A first reason is that the search window is reduced at the finer levels, and thus also the number of potential false matches per feature decreases. Even more, less computational effort is required at the coarser level to compute a first estimate for the global image deformation. At the first level of an image pyramid, random sampling is applied to estimate a set of particles which encode the posterior of the two view relation. By synthesizing powerful statistical techniques, the posterior distribution at coarser scales level can be used as an importance sampling function to draw samples from the posterior distribution at the finer level.

The *Progressive Sample Consensus (PROSAC)* algorithm [Chum and Matas, 2005] selects it samples from progressively larger sets of top-ranked correspondences. Whereas RANSAC treats all entries equally and randomly draws samples uniformly from the full dataset, PROSAC can benefit from a linear ordering defined on the set of correspondences by a similarity function used in establishing putative correspondences. Mostly, the similarity measure is readily available as the similarity score of the descriptors. If the similarity measure can predict the correctness of a match better than random guessing (e.g. by assuming that the putative correspondences with higher similarity are more likely to be inliers), PROSAC could largely reduce the computational efforts during matching. In practice the solution should typically be obtained early from a rather small set of top ranked correspondences. In the worst case, it converges toward the performance obtained by RANSAC. One difficulty is the estimation of the growth of the top ranked set from which to draw the samples. Another problem is the definition of a stopping criterion regarding the optimality of the solution.

Randomized RANSAC (R-RANSAC) [Matas and Chum, 2004] starts from the idea that most of the evaluated hypotheses are contaminated by outliers. These bad models can be rejected when a statistical test on a small number of data points reveals the presence of an outlier. If the test indicates that the hypotheses is good, it could lead to the

optimal solution with maximal support. For the *Optimized R-RANSAC* [Chum and Matas, 2008], an optimal hypothesis quality evaluation procedure (SPRT), based on WALD's sequential decision making theory is presented that returns as fast as possible on average a solution with a priori defined confidence. The use of the verification test results in a considerable speedup when compared to regular RANSAC.

The recent efforts to increase the efficiency of the standard RANSAC approach are twofold. Some optimize the process of model verification (such as R-RANSAC), others look into the sample selection process to generate potentially more useful hypotheses (e.g. PROSAC). Raguram *et al.* [Raguram et al., 2008] present a classification and evaluation of the various RANSAC-like approaches. From the evaluation they derive the useful properties of other algorithms for their *Adaptive Real-time Random SAmpling Consensus* (ARRSAC) framework. ARRSAC is suited for real-time applications, and still robustly determines the estimation by adapting to the contamination level of the data, while not making limiting assumptions.

Recently, the Optimized R-RANSAC method was extended with uncertainty estimation to *Cov-RANSAC* in [Raguram et al., 2009]. This method takes knowledge about the uncertainty of the localization of the feature points into account in the verification process. This step actually helps in terminating the RANSAC process faster, particularly when the contamination on the data is high. The underlying reason is that now all inliers should fall into the set of potential inliers, as the process explicitly takes the uncertainty into account. Thus, a correct verification set is found earlier. This approach is combined with a SPRT procedure in the verification stage: models which are likely to be uncontaminated should survive longer in the SPRT. Once a promising set of potential inliers is found, an inner RANSAC loop with uncertainty reasoning is applied. The final set of *true* inliers can now be determined much faster, as the inlier ratio now is much larger, resulting in additional computational savings.

Guided Sampling (GUNSAC) [Tordoff and Cipolla, 2005; Tordoff and Murray, 2002] also considers the modeling of localization measurement uncertainty in the estimation process by explicitly using Gaussian assumptions and linearizations. They replaced random sampling with a sampling guided by priors on the data points, which directs the search toward more valid hypotheses. The adequate handling of uncertainty sources also leads to fewer hypotheses that must be verified for a given confidence level in the estimation process.

3.3.3 Conclusion

In this overview, we can see the importance of a faster converging method than the original RANSAC. The trend is to use an approximate estimate for the model based on a few samples, e.g., on a coarse scale or by appearance similarity, or by statistical modeling, from which the search space can be reduced. A good estimation for feature RoIs is thus of importance. Another recurring item is to not only give a robust estimate of the model, but also give an indication of how consistent each sample is with the model.

The standard RANSAC approach method takes the exact position of the features into account. If the location of the features in the original sample is heavily corrupted, then the transformation model computed for that sampled subset could also be corrupted. If we model the positional uncertainty of the features, we can incorporate that knowledge into the RANSAC estimation process.

Recently, there has been some effort to include positional uncertainty into the RANSAC process, based on a localization uncertainty model with covariance matrices. We will present a similar idea, but our framework considers convex polygons and polytopes to model uncertainty in the parameter space. The matching procedure will benefit from an uncertainty approach, as the termination condition is met faster. The transformation model derived for a corrupted sample incorporates localization uncertainty, and thus an adequate consensus set is expected to be found earlier in the procedure.

3.4 Conclusion

At the moment, there are two main approaches that are often combined to match features from distinct sets.

- The local image information content is collected in a *descriptor*, which can then be matched according to some distance measure.
- The features are matched by *geometric consistency* requirements.

This chapter presents an overview of commonly used descriptors, and illustrates how to use them in image processing applications. Establishing an (initial) set of correspondences is not an easy task, therefore a discriminative and meaningful description for the extracted features must be found. The richer the local information content used by

the local feature descriptors, the easier it should be to compute the set of (putative) correspondences.

A robust transformation model fitting procedure is used to distinguish inliers from outliers in the set of putatively matched features. The research in this PhD focuses on the demand for geometric consistency, therefore most attention in the remainder of this manuscript is given to the transformation models that restrict the relations between corresponding features.

Part of our work claims that a matching procedure can benefit from requiring both spatial and parametric consistency for transformations. Now most methods only demand consistency in the image domain, and not in the parameter domain. However, the best procedure in our opinion, would be to combine both approaches to obtain the best result. Chapter 8 introduces our strategy for computing consistent matches for even very small candidate correspondence sets, after the introduction of the necessary tools in the next chapters.

Chapter 4

Geometric Transformations

Matching for spatial consistency involves the robust estimation of a transformation from the feature sets in each image. This chapter introduces the commonly used transformations in computer vision applications these days. The application at hand defines the complexity of the transformation model to use.

4.1 Introduction

Different types of geometric transformations play an important role in many computer vision applications. Before introducing the *uncertainty* of transformations in the next chapter, we first consider the computation and properties of exact unique transformations in more detail.

The choice of transformation type depends on the requirements of the specific application. Below, we will briefly sketch some of the computer vision applications in which either of the following could be applied:

- the 2D affine transformation;
- the 2D projective transformation or homography;
- the 3D-2D perspective transformation (with the pinhole camera model), and the transformation described by the fundamental matrix.

Each type of transformation allows a different degree of freedom on the relations between features in both images. Figure 4.1 gives a simple

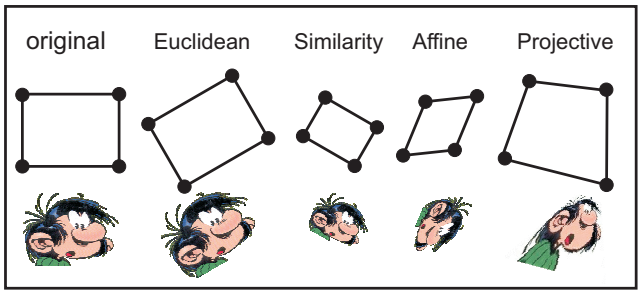


Figure 4.1: Illustration of the effect of different transformation types on a rectangle and an image (Original image of Guust Flater by Franquin [Franquin, 1997]).

indication of the influence of different transformation types on rectangles, while Figure 4.2 presents some more practical examples.

The perspective transformation is applied to map 3D coordinates to the 2D image plane (section 4.4). The pinhole camera model is widely used to describe the image acquisition process. Every two images for different camera position and orientation are related by the epipolar geometry. The fundamental matrix describes how each point in the first image is mapped onto a line in the second image. These techniques are commonly applied in stereo vision or 3D reconstruction.

Every two perspective images of the same planar surface in a 3D scene are related by a 2D homography (section 4.3). It describes the perceived geometry of objects in a plane when the point of view of the observer changes, as illustrated in Figure 4.2. A homography also relates two images of a generic scene after pure camera rotation without translation. There is a practical use for the homography in many computer vision applications, for example registration, panoramic stitching, rectification, camera calibration, virtual view rendering, etc. One can also derive information about the camera motion and apply it, e.g., in navigation and ego-localization applications.

The affine transformation is a simpler, and yet sufficiently appropriate model for image displacements under restricted circumstances (section 4.2). For example, if the transformation must be computed for smaller image regions, or when the image is acquired with a large focal length, an affine approximation suffices to model the geometric transformation. The affine transformation is for instance useful to correct uniformly distorted images after perspective distortions in satellite imaging, where geometrically correct maps are desired.

4.2 Affine Transformations

The affine transformation is a non-singular linear transformation of inhomogeneous coordinates followed by a translation. In general, an affine transformation can be understood as a combination of rotation and non-isotropic scaling, followed by translation. Its geometric effects include (combinations of) rotation, reflection, shear, scaling, similarities, translation, etc.

4.2.1 Affine Transformations of Points

The 2D affine transformation A can be defined as a map from a point $\mathbf{x}$ in $\mathbb{R}^2$ to another point $\mathbf{x}'$ in $\mathbb{R}^2$. The coordinates of a point $\mathbf{x}$ in 2D can be expressed by its Cartesian (x, y) coordinates in Euclidean space. A common practice in affine and projective geometry is to describe point locations by their homogeneous coordinates. The homogeneous coordinates of a point in a 2D projective space are usually given as $\mathbf{x} = (x_1, x_2, x_3)$, with $x = x_1/x_3$ and $y = x_2/x_3$. Every multiple (cx_1, cx_2, cx_3) then denotes the same point, for any non-zero choice of scalar c.

The affine transformation can be written as a matrix operation $\mathsf{A} : \mathbf{x} \rightarrow \mathbf{x}'$ on the point $\mathbf{x} = (x, y, w)^T$, giving the point $\mathbf{x}' = (x', y', w')^T$:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}. \tag{4.1}$$

where 6 parameters define the affine transformation: $a_1, \ldots, a_6$. Since there are 6 degrees of freedom, 3 point correspondences $(\mathbf{x_i}, \mathbf{x_i'})$ suffice to compute A as they each impose 2 constraints on the transformation parameters. For instance, if a point (x, y) in the first image corresponds to (x', y') in the second, the following constraints are imposed on the parameters $a_1, \ldots, a_6$,

$$\begin{cases} x' &= a_1 x + a_2 y + a_3 \\ y' &= a_4 x + a_5 y + a_6 \end{cases}. \tag{4.2}$$

There is no unique solution for A when the correspondences are chosen from a configuration of 3 collinear points. When a configuration does not determine a unique solution for a particular class of transformations, it is denoted as *degenerate*. A minimal configuration that does result in a unique solution is said to be *in general position*.

When more than 3 point pairs are given, the system is over-determined. Then for example a least squares solution for a non-exact mapping from the points in the first image to those in the second image can determine A.

Instead of using the algebraic description of a transformation, i.e., matrices acting on coordinates, a transformation can be described in terms of the properties or quantities that are preserved or remain invariant, denoted as the invariants. An *invariant* of a geometric transformation remains unaffected by that transformation.

Any affine transformation preserves collinearity and ratios of distances, but does not necessarily preserve angles or lengths (as observed in Figure 4.1). Several invariants for the affine transformation are the parallelism of lines, the length ratio of parallel line segments, and the ratio of areas.

A special case of the affine transformation is the similarity transformation, an orientation preserving isometry combined with isotropic scaling s,

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{4.3}$$

As this transformation has 4 degrees of freedom, it can be computed from 2 point correspondences. In this case, angles between lines are an invariant, in addition to the invariants of the general affine transformation (as illustrated in Figure 4.1). This type of transformation can be applied when no skew is expected, like in the first example in Figure 4.2. An even more restricted transformation is the Euclidean transformation, which allows only rotation and translation.

4.2.2 Affine Transformations of Straight Lines

If the equation of a straight line is written as $px+qy+r = 0$, then we can represent this line by a parameter point $\mathbf{l} = (p, q, r)$ in $\mathbb{R}^3$. All parameter points on the ray $\gamma(p, q, r) = (\gamma p, \gamma q, \gamma r)$, with $\gamma \neq 0$, represent the same line. Only the 2 independent ratios of the 3 parameters are of importance, so that a line actually has 2 degrees of freedom. These parameters are e.g. the slope and the y-intercept of the line.

If a non-singular affine transformation A (4.2) transforms points as

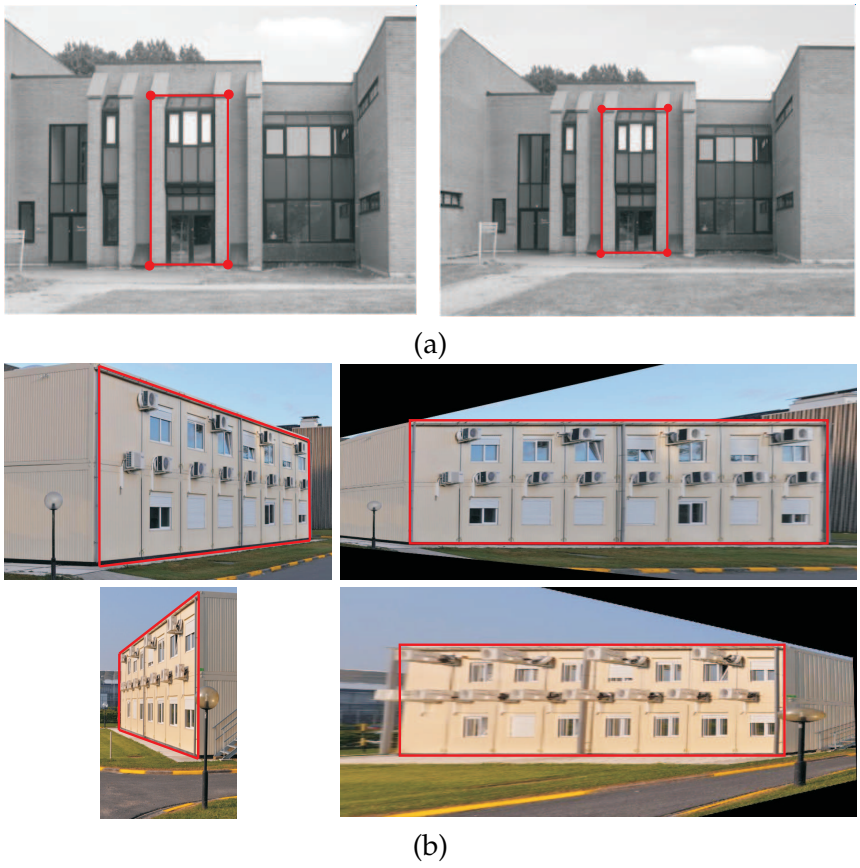


Figure 4.2: Illustration of the different effects of affine and projective transformations. (a) Both pictures are taken with a different zoom factor, and from a slightly different camera view point. Because of the larger focal length by which both images are acquired, the transformations can be modeled as affine, or even as a similarity transformation. (b) When perspective effects come into play, a homography is necessary to rectify rectangular image regions.

$\mathbf{x}' = \mathsf{A}\mathbf{x}$, then lines are transformed as $\mathbf{l}' = |\mathsf{A}|\mathsf{A}^{-T}\mathbf{l} = \mathsf{B}\mathbf{l}$, or

$$\begin{bmatrix} p' \\ q' \\ r' \end{bmatrix} = \begin{bmatrix} a_5 & -a_4 & 0 \\ -a_2 & a_1 & 0 \\ a_2a_6 - a_3a_5 & a_3a_4 - a_1a_6 & a_1a_5 - a_2a_4 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{4.4}$$

where B is the transpose of the adjugate of A. The latter is the inverse of A, multiplied with the determinant of A. Note that the transformation of the line parameters is linear.

For a given vector $\mathbf{l}$ of line parameters, the set of all nonzero scalar

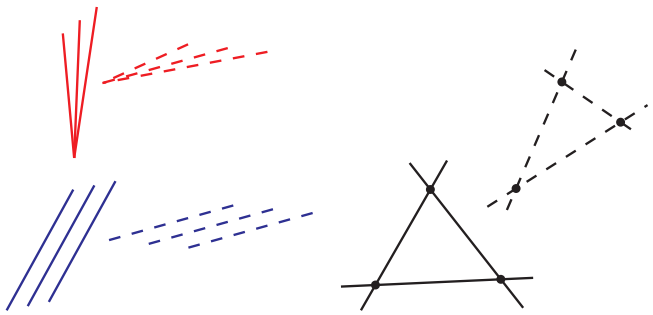


Figure 4.3: (a) Degenerate configurations for an affine line transformation: 3 concurrent or 3 parallel line correspondences. (b) Computing the affine transformation matrix for 3 line correspondences is equivalent to computing the transformation matrix for 3 point correspondences, i.e., the intersection points of each line pair.

multiples of transformed line parameters (4.4) is denoted as $\mathsf{A} < \mathbf{l} >$.

For computing the transformation matrix B, 3 line correspondence pairs are required as the matrix has 6 degrees of freedom. There are degenerate cases, as illustrated in Figure 4.3 (a) for 3 concurrent lines or for 3 parallel lines. Then, 1 degree of freedom remains, and the affine transformation cannot be determined exactly.

When 3 lines and their transformed images are given, this problem can be easily reduced to the problem where 3 points and their images are given, provided the lines are in general position. It suffices to find the affine transformation which maps the 3 intersection points of the 3 lines on the image intersection points of the transformed lines (as illustrated in Figure 4.3 (b)).

4.2.3 Points and Lines

The combination of line and point features could prove beneficial in the computation of transformations, e.g., in cases where only few reliable features are detected. The affine transformation can be specified by n points (x_i, y_i) and m lines $y = xp_j + q_j$, with $n + m = 3, n, m \geq 0$, and their images, when the points are not located on the lines.

At first sight, Eq. 4.4 leads to non-linear equations for the transformation parameters $a_1, \ldots, a_6$. Linear equations can be obtained by using points on the lines instead of the line parameters. The exact lo-

cation of the image points must not be specified, only the requirement that the image points must lie on the image line is used. This is one of the assets of using a combination of points and lines as features.

Since at least one line $\mathbf{l}$ is known, we can select two arbitrary points on $\mathbf{l}$. For example, $px + qy + r = 0$ contains the points $(0, -r/q)$ and $(-r/p, 0)$ (provided the line is not parallel with the horizontal or vertical axis).

The images of both points under A (4.1) are the points $(-a_2 r/q + a_3, -a_5 r/q + a_6)$ and $(-a_1 r/p + a_3, -a_4 r/p + a_6)$. If we substitute the transformed points into the equation of the transformed line $\mathbf{l}'$, $p'x' + q'y' + r' = 0$, we find

$$\begin{array}{rcl} (-a_2 r/q + a_3)p' + (-a_5 r/q + a_6)q' + r' & = & 0 \\ (-a_1 r/p + a_3)p' + (-a_4 r/p + a_6)q' + r' & = & 0 \end{array} \tag{4.5}$$

which are linear equations in the unknowns $a_1, ..., a_6$.

For horizontal or vertical lines, either p or $q = 0$. In that case, division by zero occurs in Eq. 4.5. This can be avoided by choosing another point on the line.

Another consideration is that the configuration must be in general position, which is the case for a combination of 2 lines and 1 point, but not for 2 points and 1 line. In the latter configuration one degree of freedom remains unspecified.

4.3 Projective Transformations

A projective transformation, also denoted by homography, in $\mathbb{P}^2$ is any non-singular linear transformation from points to corresponding points in homogeneous coordinates that maps straight lines to straight lines in another viewpoint. Thus incidence and cross-ratio relations are preserved, but not (necessarily) parallelism of lines. A homography is commonly used to describe the relation of points (in homogeneous coordinates) on the same 3D world plane in distinct perspective camera images.

4.3.1 Point Homographies

A projective transformation is seen as any invertible linear transform of homogeneous coordinates. When a point $\mathbf{x}$ is represented by the

3-vector of homogeneous coordinates $(x, y, w)^T$, the projective transformation H is a mapping of points $\mathbf{x}$ in a projective plane $\mathbb{P}^2$ to points $\mathbf{x}' = (x', y', w')^T$ in $\mathbb{P}^2$ by an invertible 3×3 matrix H, i.e., $\mathbf{x}' = \mathsf{H}\mathbf{x}$:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}. \tag{4.6}$$

H is a homogeneous matrix, since only the ratio of the matrix elements is important: any multiple $\alpha\mathsf{H}$ of H with $\alpha \neq 0$ describes the same transformation, just as all multiples $\beta(x', y', w'), \beta \neq 0$ describe the same point. Therefore H can be normalized by setting one of its elements to 1 and scaling the others appropriately.

Since H has 8 degrees of freedom, 4 point-to-point correspondences establish enough linear equations to compute the entries of H, up to an insignificant multiplicative factor. As each correspondence pair imposes 2 linear equations on the elements of H, we obtain a closed solution for H by solving the system for 4 correspondence pairs. The only restriction is that the points must be in general position, i.e., no 3 points can be collinear. Otherwise dependencies would occur among the 8 equations.

Hartley and Zisserman [Hartley and Zisserman, 2003] present several computation methods for the transformation matrix H for an overdetermined system (i.e. when more than 4 point pairs are given). For example, a least squares solution for a non-exact mapping from the points in the first plane to those in the second plane determines H.

The homography does not preserve size or angle, but does preserve incidence, i.e., any two distinct lines meet in a unique point. Also parallel lines in a perspective image converge in a unique point, the vanishing point. When rectangular planes in perspective images are rectified, i.e., a homography is applied so that its lines become parallel in the resulting image, the vanishing point is a point at infinity. A point at infinity is represented in homogeneous coordinates (x, y, w) with $w = 0$.

The projective transformation can be thought of as a generalization of the affine transformation, where the parameters h_7 and h_8 account for its perspective effects. Or vice versa, an affine transformation is a special type of homography where the parameters $h_7 = h_8 = 0$, and $h_9 = 1$. Affine transformations do not map any objects from the affine space to the plane at infinity, or conversely.

4.3.2 Projective Transformation of Straight Lines

If the points $\mathbf{x_i}$ lie on the line with parameter vector $\mathbf{l}$, then the transformed points $\mathbf{x'_i}$ lie on the line

$$\mathbf{l'} = \mathsf{H}^{-T}\mathbf{l}, \tag{4.7}$$

given a point transformation $\mathbf{x'} = \mathsf{H}\mathbf{x}$, with H of the form (4.6). H^{-T} is the transpose of the inverse of H.

If the projective transformation H is non-singular, the line parameters $\mathbf{l} = (p, q, r)^T$ are transformed to $\mathbf{l'} = (p', q', r')^T$ with the linear transformation $\mathsf{G} = |\mathsf{H}|\mathsf{H}^{-T}$ as

$$\begin{array}{rcl} p' & = & -h_6h_8p + h_5h_9p + h_6h_7q - h_4h_9q - h_5h_7r + h_4h_8r \\ q' & = & h_3h_8p - h_2h_9p - h_3h_7q + h_1h_9q + h_2h_7r - h_1h_8r \\ r' & = & -h_3h_5p + h_2h_6p + h_3h_4q - h_1h_6q - h_2h_4r + h_1h_5r \end{array} \tag{4.8}$$

In the computation of the transformation parameters, one must select a non-degenerate configuration of 4 line feature correspondences, i.e., 4 non-concurrent lines. The configuration of 4 parallel lines is also a degenerate configuration, as these meet in a point at infinity.

The robust computation of transformation parameters from small line correspondence sets requires additional care regarding the normalization procedure, as the difference in magnitude of line parameters can be large [Dubrofsky and Woodham, 2008].

4.3.3 Points and Lines

The combination of line and point features could increase the reliability of transformation computations, certainly in the cases where few features are used. Every feature (point or line) correspondence pair gives 2 constraints on the transformation H. So the correspondence of m point pairs and n line pairs, with $m + n = 4$, is sufficient to solve for H.

We can deduce a set of linearized equations for the line transformation parameters along the same lines as in Section 4.2.3, resulting in

$$\begin{array}{rcl} (h_3p'q + h_6qq' - h_2p'r - h_5q'r + h_9qr' - h_8rr')/q & = & 0 \\ (h_3pp' + h_6pq' - h_1p'r - h_4q'r + h_9pr' - h_7rr')/p & = & 0 \end{array} \tag{4.9}$$

which are linear equations in the unknowns $h_1, ..., h_9$.

The cases where $m + n = 3 + 1$ or $1 + 3$ are geometrically equivalent to the case with either 4 line pairs or 4 point pairs. That is, respectively,

Figure 4.4: Illustration of the geometric equivalence of point-line configurations [Hartley and Zisserman, 2003]. A configuration of 2 points and 2 lines is equivalent to 5 lines with 4 concurrent, or 5 points with 4 collinear.

either 1 given line and 3 lines defined by connecting every 2 of the 3 given points, or 1 point and 3 intersection points of every 2 of the 3 given lines.

However, one cannot combine 2 point and 2 line correspondences. Because this configuration is geometrically equivalent to a configuration with 5 lines of which 4 are concurrent, or 5 points of which 4 collinear. A quick sketch illustrates this configuration in Figure 4.4. This configuration is degenerate, i.e., it does not determine a unique solution for this particular transformation, as the system will not be of full rank in that case. If the configuration is mapped onto a corresponding configuration, the homography is not sufficiently constrained, and there exists a one-parameter family of homographies mapping the two-point and two-line configuration to the corresponding configuration [Hartley and Zisserman, 2003].

4.4 Perspective Transformations and the Epipolar Geometry

Each camera yields a mapping from the 3D world to a 2D image. The conversion from the 3D world to a 2D image is commonly referred to as a perspective projection, and can be described by a camera model.

4.4.1 Pinhole Camera Model

The widely used generalized pinhole camera model [Hartley and Zisserman, 2003] describes the projection of the 3D world onto a 2D image. The projection is modeled by incident rays, passing through the camera center point $\mathbf{C}$. Each ray then corresponds to a single point $\mathbf{x}$ in the image. The operation is illustrated in Figure 4.5.

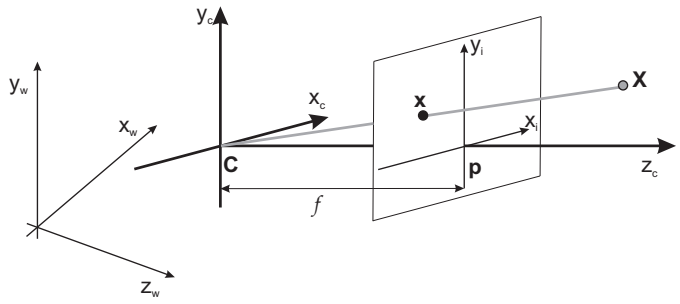


Figure 4.5: Pinhole camera model: each world point $\mathbf{X}$ is projected onto a point $\mathbf{x}$ in the image plane by a ray through the camera center $\mathbf{C}$. $\mathbf{C}$ is denoted as the center of projection. The ray perpendicular to the image plane through $\mathbf{C}$ is the optical axis, and it meets the image plane in the principal point $\mathbf{p}$. The distance between $\mathbf{C}$ and $\mathbf{p}$ is the focal length f. In real cameras, the image plane is actually behind the camera center, and produces a mirrored image. Here the projection problem is simplified by placing a virtual image plane in front of the camera center to produce an unmirrored image.

If we use a homogeneous 4-vector $\mathbf{X} = [X, Y, Z, 1]^T$ for a world point, and represent an image point by the homogeneous 3-vector $\mathbf{x} = [x, y, w]^T$, the camera projection is modeled by a 3×4 homogeneous projection matrix P as $\mathbf{x} = \mathsf{P}\mathbf{X}$.

The optical imperfections of a real CCD camera are incorporated as the *internal camera parameters* in a camera calibration matrix K. P also describes the *external camera orientation and position* with respect to objects in the 3D world in R and $\mathbf{C}$. Then

$$\mathbf{x} = \mathsf{P}\mathbf{X} = \mathsf{K}\mathsf{R}[\mathsf{I}| - \mathbf{C}]\mathbf{X}, \tag{4.10}$$

with $[\mathsf{I}| - \mathbf{C}]$ the identity matrix augmented with $-\mathbf{C}$.

External: One must relate the camera location and orientation to the location of points in 3D space, i.e., to the world reference frame. Figure 4.5 illustrates how the origin of the Euclidean camera frame must be positioned in the Euclidean world coordinate frame. Both frames are related through a rotation and a translation, where the 3-vector $\mathbf{C}$ denotes the camera coordinates in the world frame, and a 3×3 rotation matrix R represents the orientation of the camera frame with respect to the world frame.

Let $\mathbf{X}$ and $\mathbf{X}_c = [X_c, Y_c, Z_c]^T$ represent the same point in respectively the world and the camera frame, then $\mathbf{X}_c = \mathsf{R}[\mathsf{I}| - \mathbf{C}]\mathbf{X}$.

Internal: The internal orientation K of the camera is of the form

$$\mathsf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

where

- α_x and α_y are the scale factors in the x- and y- coordinate direction;
- s is the skew of the camera pixel elements (which will be zero for most normal cameras);
- (x_0, y_0) are the coordinates of the principal point $\mathbf{p}$.

K relates points in the camera frame to a 2D projection as $\mathbf{x} = \mathsf{K}\mathbf{X}_c$.

In CCD cameras the pixels can be non-square and skew. Therefore, we introduce unequal scale factors in each direction. If the number of pixels per unit distance in image coordinates are m_x and m_y, then $\alpha_x = fm_x$ and $\alpha_y = fm_y$, respectively the camera focal length in pixel dimensions in x- and y-direction. In a similar way, we can give the principal point in terms of pixel dimensions as $x_0 = m_x p_x, y_0 = m_y p_y$, where p_x, p_y are the coordinates of the principal point $\mathbf{p}$.

4.4.2 Epipolar Geometry

In general, it is not possible to directly find a point-to-point relation between the projections of the same world point $\mathbf{x} = \mathsf{K}_1\mathsf{R}_1[\mathsf{I}| - \mathbf{C_1}]\mathbf{X}$ and $\mathbf{x}' = \mathsf{K}_2\mathsf{R}_2[\mathsf{I}| - \mathbf{C_2}]\mathbf{X}$, for a different camera position and orientation. As illustrated in Figure 4.6, establishing such a relation requires knowledge about the distance, or the depth, of the point $\mathbf{X}$ on the emanating ray $\mathbf{C_i} \rightarrow \mathbf{x}$ to the imaging plane.

When two cameras observe a 3D scene from two distinct positions, there are a number of geometric relations that lead to constraints for corresponding 2D image points. These relations are derived based on the assumption that the cameras can be approximated by the pinhole camera model. It is then possible to define a relation for the corresponding points in both images. A correspondence for a point in the first image must be found on the projection of its emanating ray onto the second image. A more formal definition of this statement is given by the *epipolar geometry* of images.

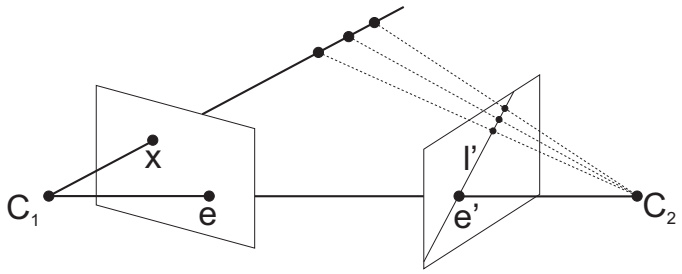


Figure 4.6: The depth of a 3D world point $\mathbf{X}$ to the image plane along the ray is generally not known. Then the epipolar geometry describes how to find the line $\mathbf{l}'$ in the second image on which the correspondence for $\mathbf{x}$ must be located.

If the viewpoint of the camera changes in between two consecutive capture instants, or if two different cameras are used, both camera centra are distinct. Each center projects onto a different point in the other camera's image plane. This projection is denoted as the epipole, indicated by $\mathbf{e}$ and $\mathbf{e}'$ in Figure 4.6. There is a single 3D line (the baseline) connecting both epipoles and both camera centra.

When two cameras observe the same point $\mathbf{X}$ in the 3D scene, the emanating ray $\mathbf{C_1} \rightarrow \mathbf{x}$ is seen by the second camera as a line in its image plane, i.e., the epipolar line through its epipole $\mathbf{e}'$. Symmetrically, the ray $\mathbf{C_2} \rightarrow \mathbf{x}'$ is captured by the first camera as an epipolar line through the epipole $\mathbf{e}$.

For each point $\mathbf{x}$ in the first image, the epipolar line $\mathbf{l}'$ in the second image can be computed. The observation $\mathbf{x}'$ of $\mathbf{X}$ is projected on the epipolar line $\mathbf{l}'$. This is denoted as the *epipolar constraint* for corresponding image points, which is described by the *fundamental matrix* F. The parameters of the epipolar lines are given by $\mathbf{l}' = \mathsf{F}\mathbf{x}$, or symmetrically $\mathbf{l} = \mathsf{F}^T\mathbf{x}'$. Thus, for corresponding points $\mathbf{x}$ and $\mathbf{x}'$ in the two cameras, $\mathbf{x}'$ is located on the epipolar line $\mathbf{l}'$,

$$\begin{aligned} \mathbf{x'}^T\mathbf{l}' &= 0 \\ \mathbf{x'}^T\mathsf{F}\mathbf{x} &= 0 \\ \begin{bmatrix} x' & y' & w' \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} &= 0. \end{aligned} \tag{4.12}$$

The fundamental matrix F has 7 degrees of freedom. The common scaling of the elements in the 3×3 homogeneous matrix F is not significant. In addition, F must also satisfy the constraint $\det(\mathsf{F}) = 0$, which

again removes 1 degree of freedom. As a result, we can say that F can be computed from at least 7 point correspondences. There are many algorithms in literature to retrieve F from a set of point correspondences, e.g., the Longuet-Higgins method [Longuet-Higgins, 1987], or Hartley's normalized 8-point algorithm [Hartley and Zisserman, 2003], to name a few. In Figure 4.7, we give an example of the epipolar geometry for two images of the same scene.

As an alternative, the fundamental matrix F can be computed from known camera parameters (e.g. when the motion of the camera between 2 captured frames, or the relative position and orientation for 2 distinct cameras, is known). Both camera matrices can then be defined as $\mathsf{P}_1 = \mathsf{K}_1[\mathsf{I}|\mathbf{0}]$ and $\mathsf{P}_2 = \mathsf{K}_2[\mathsf{R}|\mathbf{t}]$ (with $\mathbf{t} = -\mathsf{R}\mathbf{C}$), and the fundamental matrix is given by

$$\mathsf{F} = {\mathsf{K}_2}^{-T}[\mathbf{t}]_X \mathsf{R} {\mathsf{K}_1}^{-1}, \tag{4.13}$$

with $[\mathbf{t}]_X$ the skew-symmetric matrix for $\mathbf{t}$.

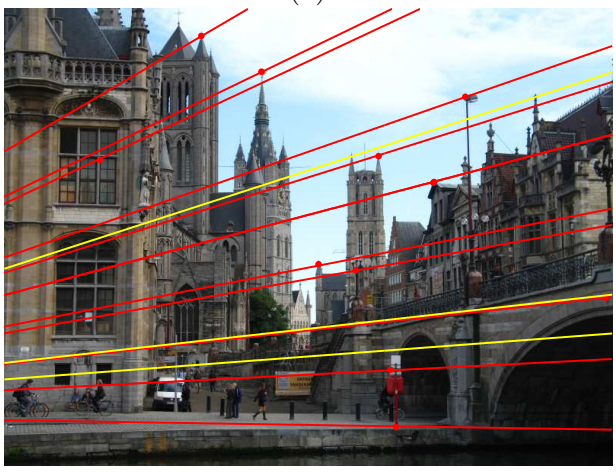
4.4.3 The Epipolar Geometry for Lines

There is no unique correspondence for parameterized lines (thus of infinite length) in the epipolar geometry, as a line is back projected onto a plane in 3D space. Its image covers the whole image plane of a camera, except for some special cases.

There has been some research to compute the transformation of line segments of finite length. Only a weak overlap constraint can be given for line segments by applying the epipolar constraint to its two end points. Mostly these methods involve matching strategies based on geometric properties, like orientation, length, overlap, etc. Considering the nearest neighbor appears to be a good strategy for line tracking applications. More information and less ambiguity are provided by matching sets of line segments, mostly by graph-matching techniques, at the cost of an increase in complexity.

Schmid and Zisserman present a line-matching scheme where the epipolar geometry is applied to facilitate the automated line matching by computing a cross-correlation based score for putative line correspondences [Schmid and Zisserman, 1997]. In [Schmid and Zisserman, 2000b], they show how to establish relations for the geometry of imaged lines and curves in two or more views, by applying a pencil of planes, or homographies, to transfer curves from one image to another.

(a)



(b)

Figure 4.7: An example of the epipolar geometry for two images from the same scene, but observed from different position. $(a - b)$ The red dots denote the set of points for which the fundamental matrix F is computed using the normalized 8-point algorithm [Hartley and Zisserman, 2003]. The epipolar lines (red lines in (b)) are computed as $\mathbf{l}' = \mathsf{F}\mathbf{x}$ for each point of the initial set (red dots in (a)). Correspondences for points not in that initial set (yellow dots in (a)) can be found on their corresponding epipolar lines (yellow lines in (b)).

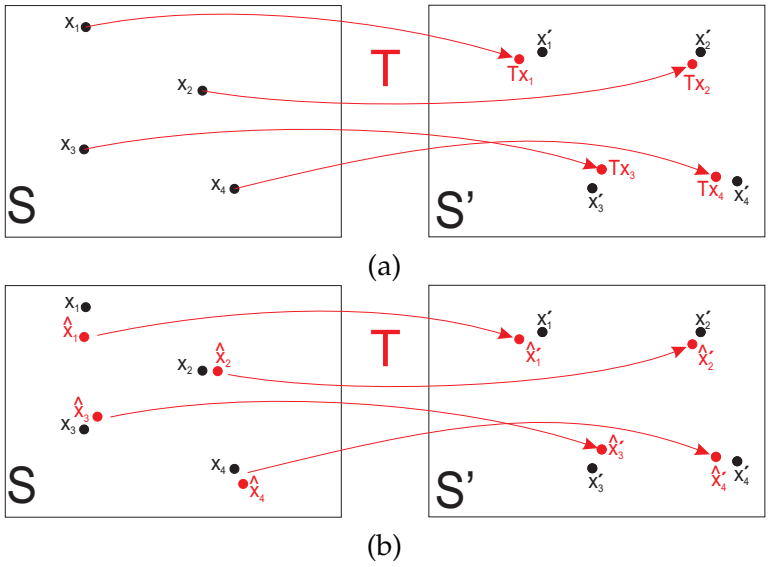


Figure 4.8: Geometric cost criteria in the determination of an optimal transformation: (a) the transfer error and (b) the reprojection error [Hartley and Zisserman, 2003].

4.5 Uncertainty

The computation of the transformation parameters requires a solution for a linear system $\mathsf{A}\mathbf{x} = \mathbf{b}$. In most applications, the transformation model with n unknown parameters is computed from a set of m correspondences (with A an $m \times n$ matrix). If $m > n$, an exact solution does not exist in most cases, due to inexact measurements of the location, or noise. However, it still makes sense to seek an estimate for $\mathbf{x}$ that is closest to providing a solution for the over-determined system. Then the question that naturally arises is: what should be minimized?

Hartley and Zisserman give an overview of cost functions that can be applied in the estimation of the optimal transformation for over-determined systems [Hartley and Zisserman, 2003]. They consider both an algebraic error and a geometric distance criterion. In a least-squares solution for an overdetermined system, an approximate solution is derived by minimizing the vector norm $||\mathsf{A}\mathbf{x} - \mathbf{b}||$. The solution is then given by $\mathbf{x} = \mathsf{A}^{+}\mathbf{b}$ with A^{+} the pseudo-inverse. However, this solution has some drawbacks regarding outlier handling. The error is not adequately represented if an outlier has a larger influence on the com-

putation of the transformation parameters, than other pairs that yield rather small deviations. There are many ways of dealing with the outlier sensitivity of overdetermined systems.

As illustrated in Figure 4.8, the geometric cost may either be measured

- as a transfer error in one plane: $\sum_i d(\mathbf{x}'_i, \mathsf{T}\mathbf{x}_i)$ in the image plane, or $\sum_i d(\mathbf{x}_i, \mathsf{T}^{-1}\mathbf{x'}_i)$ in the source plane;
- as a symmetric transfer error in both planes: $\sum_i d(\mathbf{x}_i, \mathsf{T}^{-1}\mathbf{x'}_i) + d(\mathbf{x}'_i, \mathsf{T}\mathbf{x}_i)$;
- or as a reprojection error in both planes: find the transformation T that minimizes $\sum_i d(\mathbf{x_i}, \hat{\mathbf{x}}_\mathbf{i})^2 + d(\mathbf{x}'_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i})^2)$ with $\hat{\mathbf{x}}'_\mathbf{i} = \mathsf{T}\hat{\mathbf{x}}_\mathbf{i}$, a problem that is commonly solved by an expectation-maximization algorithm.

We will present a geometric criterion apt for our framework that rather considers the maximal error, in chapter 6.

4.6 Conclusion

This chapter presented different transformation types commonly applied in image processing techniques regarding correspondence problems. For each transformation type we also emphasized the degenerate configurations to avoid in the selection of sample sets in robust estimation procedures.

The properties of the transformations are exploited in the definition of transformation uncertainty. The next chapter will introduce our geometric uncertainty framework which explicitly takes the feature localization uncertainty into account in the computation of the parameters for affine and projective transformations.

Chapter 5

Uncertainty

Varying conditions in digital image acquisition invariably lead to uncertainty about the occurrence, location and shape of image features. The uncertainty about detected feature locations can be modeled by defining a set of possible locations, without knowing its exact position. Suppose an image can be mapped onto another image according to a given transformation. In standard geometry this requires one-to-one relationships between both point sets. We propose a geometric uncertainty framework to model the uncertainty about feature localization and its propagation into geometric transformations.

5.1 Introduction

Uncertainty is inherent to the imaging process. Uncertainty about the occurrence, location and shape of image features is inevitable because of quantization effects, varying illumination conditions, geometric distortion and noise in digital image acquisition, as discussed in Section 2.4. For example, due to lens distortion, points can shift up to tens of pixels away from their actual position. The digitization process introduces a localization error of 0.5 times the pixel size. Also the image processing steps could introduce localization errors, e.g., small errors made by feature detectors because of noise, or after subsampling, etc.

All possible sources of feature uncertainty propagate through the geometric computation chain and affect the accuracy of the output result, like for feature transformations. Coping with feature uncertainty

is therefore one of the major challenges in computer vision applications that want to establish correspondences between features in distinct images, that want to derive geometric relations between features, or that involve geometric reasoning. A good mathematical model for feature uncertainty can greatly improve the performance of a correspondence algorithm.

The following section describes the common approach to deal with geometric uncertainty in computer vision. In this approach, geometric uncertainty about feature location is mostly modeled by covariance matrices and uncertainty ellipses. The next sections introduce our uncertainty model in more detail (5.3), both for affine transformations of both point (5.4) and line (5.5) features.

Our uncertainty model of polygonal regions for the representation of the uncertain feature localization could provide more flexibility and offer some additional advantages. The localization uncertainty will translate into transformation uncertainty in a natural way when determining correspondences. The transformation uncertainty can be represented as a polytope in the parameter space, i.e., a convex bounded subspace which can be described as a system of linear equations or inequalities. That allows us to develop simple solutions based on efficient geometric techniques.

The transformation uncertainty model allows to compute accurate regions of interest (RoIs) for each feature, so that a set of candidate correspondences is found quickly and reliably during the matching procedure. In our opinion, computing RoIs should be an essential part of any real-time image and video processing method that requires feature matching.

5.2 Geometric Uncertainty Modeling

In many applications concerning geometric reasoning, there is a need to deal with imprecision. Furthermore, when computing the propagation of uncertainty, one must take geometric relations and constraints into account, which is often not straightforward. Recently, the developments and literature in uncertain geometry have become more elaborate and systematic.

Whereas the common approach applies statistical geometric inference to model and propagate the localization uncertainty of features, we advocate a uniform approach for geometric uncertainty that takes

into account geometric relations. It consists in the ideal case of one coherent methodology that can be applied to many geometric problems.

5.2.1 Uncertain Geometry

Kanatani was one of the first to use statistical inference in a systematic manner, as a rigorous mathematical framework to solve uncertainty problems in geometric reasoning [Kanatani, 1996, 2004, 2005].

Kanatani regards the position of a feature as sampled from a set of possible positions and uses statistical inference to propagate the uncertainty further into the geometric reasoning chain. A measure for the spread of the potential positions of each feature point, i.e., its positional uncertainty, is given by its covariance matrix.

Kanatani and Kanazawa [Kanazawa and Kanatani, 2001] introduce an overview of different methods for the computation of covariance matrices and examine whether it really reflects the accuracy of feature locations. They have shown that the uncertainty of the feature localization can be related to the local image content. Then the uncertainty can be modeled by covariance matrices, under the assumption of a Gaussian distribution for the position of features. Figure 5.1 (a) illustrates the covariance matrices and uncertainty ellipses related to the point localization.

If the intensity variations around a feature point are almost homogeneous in all directions, the extracted feature position is somewhat ambiguous whatever algorithm is used. In other words, the detected locations then have a large spread, which is reflected in the positional covariance matrix. For a region with a nearly homogeneous intensity variation, we can think of the probability distribution as isotropic, and draw the uncertainty ellipse, i.e., a typical equiprobability line, as a large circle. For a location on an object boundary, the covariance matrix results in an elongated uncertainty ellipse along the boundary edge.

However, feature points are rarely chosen in such regions, rather in regions that show large intensity variations in multiple directions. If the intensity varies greatly around the feature point in all directions, any feature detection algorithm could accurately locate it, meaning that the probability distribution of the detected feature location is strongly peaked. Then the covariance matrices for features extracted by detection algorithms can also be regarded as nearly isotropic. This results in circular regions, but much smaller than those obtained for the more ho-

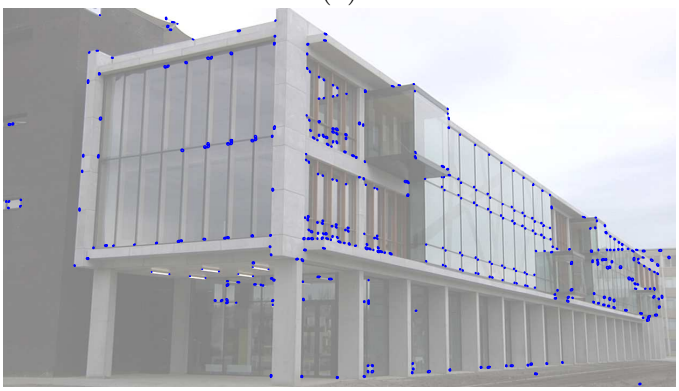
mogeneous regions. Figure 5.1 $(b-c)$ show the uncertainty ellipses for the interest points detected by the Harris corner detector, which shows that the (appropriately scaled) uncertainty ellipses are nearly isotropic.

In his work, Kanatani shows how the feature uncertainty affects the accuracy of further geometric computations, such as the detection of lines and circles, or the computation of transformations such as homographies or fundamental matrices. He adapts and applies standard statistical analysis techniques (e.g. the Akaike Information Criterion or Rissanen's Minimum Description Length) to infer geometric models from the statistics of the underlying data, i.e., the features. To make computations feasible, he assumes a Gaussian distribution for the feature locations, and needs to apply first order approximation, assuming small uncertainty, in the geometric fitting computations. Next, though Kanatani uses a homogeneous vector representation, Euclidean normalization is required for the 2D and 3D points. This step is motivated by an otherwise indefinite scaling of vectors, but doing so he loses the ability to handle points at infinity. The explicit combining of both Euclidean and homogeneous vector representations leads to awkward expressions for covariance matrices and in the error propagation.

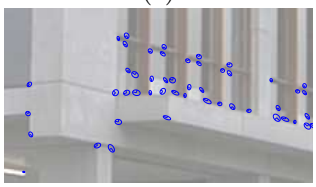
Criminisi [Criminisi, 2001] shows how accurate measurements from images can give a reliable estimate about the geometric structure of the imaged scene. He tries to gather useful geometrical information about the represented scene, starting from the detection and extraction of features, whether automatic (by edge or interest point detectors) or manual. However, the position of features can only be determined to a finite accuracy as all features extracted from an image are subject to measurement errors. These errors inevitably propagate along the computation chain, causing a loss of accuracy in the measurements, and thus in the final structure.

Criminisi considers the propagation of these errors throughout the measurements formulas by using a first order error analysis to quantify the uncertainty on the final measurements. The positional uncertainty is given as a two-dimensional Gaussian distribution, and its uncertainty is modeled by covariance matrices, for which he gives explicit expressions. From the localization uncertainty, the uncertainty on the computed homography matrices can be computed and modeled on its turn by covariance matrices. The propagation of the errors through the computations, e.g., for distance measures between computed (i.e. transformed) points, can then be written down. The results of the un-

(a)



(b)



(c)

Figure 5.1: (a) The uncertainty ellipses (i.e. an equiprobability curve) for the covariance matrices computed at some specifically chosen image locations: along edges, in corners and in nearly homogeneous image patches. (b) The uncertainty ellipses at the location of detected Harris corners (Figure 2.4). (c) A more detailed view. In $(a-b)$ the ellipses are magnified for display.

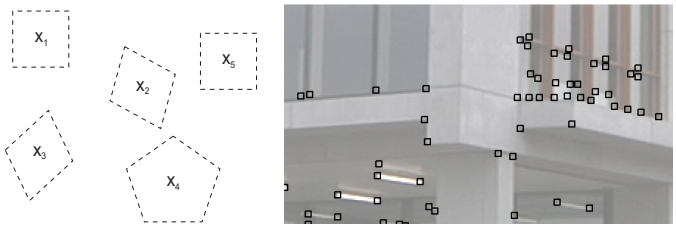
certainty analysis are validated by Monte Carlo simulations to support his approach.

First order error propagation analysis is also employed in the work of Se *et al.* [Se et al., 2002]. They take the localization uncertainty of detected SIFT landmarks in the environment into account in the computation of the trajectory of a mobile robot. The positional uncertainty of the landmarks is modeled by covariance matrices and represented by uncertainty ellipses.

In Förstners approach, geometric parameters and relations are derived by statistical inference to take geometric uncertainty into account [Förstner, 2004]. Förstner develops elegant representations for uncertainty and discusses the propagation of uncertainty in projective geometry using covariance matrices. He has worked out several simple-to-use tools that are based on statistical inference, and which can be used to analyze and propagate uncertainty through geometric reasoning chains [Förstner, 2005]. This work has been further extended to other problems [Perwass et al., 2005; Perwass and Förstner, 2006].

Förstner claims that the homogeneous vector and matrix are the best suited choice as a representation for geometric entities and transformations. Because of his opinion that estimation procedures must be easy to use, functional relations between observed quantities and unknowns must first of all be represented as generically as possible. Second, he applies Maximum Likelihood estimation based on a Gauss-Helmert model, as a generic estimation procedure, which allows to model estimation problems with any number of non-linear constraints. By exploiting the multi-linearity of the geometric constructions and constraints, he succeeds in extending his approach beyond the projective geometry, as it appears to be feasible to reformulate expressions in terms of coordinate vectors with an attached covariance matrix.

Although the above research proposes to use both statistics and projective geometry for spatial reasoning under uncertainty, some questions remain. One of the main problems left is the search for an adequate representation and procedures for geometric reasoning. Mostly, in statistics, the uncertainty of measurements is represented by covariance matrices. And generally, the first two moments of the distribution are used (instead of working with the probability densities). This appears to be adequate as long as the signal to noise ratio is high enough (e.g. 10:1). Then it is claimed that the propagation of uncertainty into the linear algebra calculus is sufficiently accurate. However, the covari-



(a) (b)

Figure 5.2: (a) An uncertainty region represents the locus of all possible measured positions of the actual point in our uncertainty scheme. (b) An illustration for a real image.

ance representation is nearly always an approximate uncertainty representation. The embedding into more fundamental algebraic concepts does not seem to be possible, but is kept as far as possible [Förstner, 2004].

5.2.2 Our Uncertainty Approach

The presented work builds on results obtained earlier in this domain of the uncertain digital geometry. In his research, Peter Veelaert, one of the promoters, presented a methodology for geometric reasoning when the position and parameter vectors are imprecise or uncertain. He proposes a simple, universal scheme to model the uncertainty on a (feature) point position [Veelaert, 1999b].

Whereas statistical geometric inference methods start from covariance matrices and uncertainty ellipses, in our approach the positional uncertainty is modeled by polygonal uncertainty regions in which features are likely to occur (Figure 5.2). That is, instead of estimating a probability density function (pdf) for a feature, we construct an uncertainty region in which the feature is likely to occur. Roughly, an uncertainty region is a region which indicates where a pdf is above a certain threshold.

An uncertain version for notions such as straightness, parallelism, collinearity and concurrency of digital lines has been developed following the same guidelines [Veelaert, 1999a, 2001, 2002]. For example in Figure 5.3, a set of pixels is considered to be a straight line (segment) if we can select in each uncertainty region a real point, so that the selected points lie on a common straight line [Veelaert, 1999b].

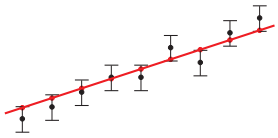


Figure 5.3: An illustration for the definition of a digital straight line segment in uncertain geometry [Veelaert, 1999b].

One of the strong points of this theory is the direct link between the uncertainty model and the graph-theoretical optimization problems. Many optimization problems in uncertain geometry can be formulated as graph algorithms such as minimum clique covering, minimum coloring or minimum dominating set problems [Veelaert, 1999b,a, 2000, 2003a]. The solution of a given optimization problem can always be interpreted in terms of uncertainty regions and parameter domains.

Our approach allows for a simple propagation of the localization uncertainty into higher level geometric reasoning. In this work, we mostly consider the effect of localization uncertainty on the computation of geometric transformations. The uncertainty in the geometric transformation space can again be modeled by polytopes instead of uncertainty ellipsoids, which provides more flexibility and offers additional advantages.

One important advantage is that a polygon or a polytope is in general a good approximation for any convex shape, e.g., a polytope must not be symmetrical, as is the case for an ellipsoid. Thus, in applications regarding positional or geometric uncertainty polytopes can provide a better description of the shape of the uncertainty region, which is an important aspect to take into account. As an example, consider the intersection of two (uncertain) straight lines, as illustrated in Figure 5.4.

A formal definition of the uncertainty of straight lines in the digital image domain is given by Veelaert in [Veelaert, 2003b, 1999b]. From this work, we derive a representation for the positional uncertainty of lines as follows. Let **l** be a line segment $px + qy + r = 0$ with slope $-1 \leq \alpha = -p/q \leq 1$ and two endpoints with distinct x-coordinates (x_1, y_1) and (x_2, y_2). If τ denotes the uncertainty parameter, then the uncertainty region for the line is the set of all parameter vectors (p, q, r) that satisfy the following system of inequalities

$$-\frac{\tau}{2} \leq y_i - (-\frac{p}{q}x_i - \frac{r}{q}) \leq \frac{\tau}{2},\ i = 1, 2, \tag{5.1}$$

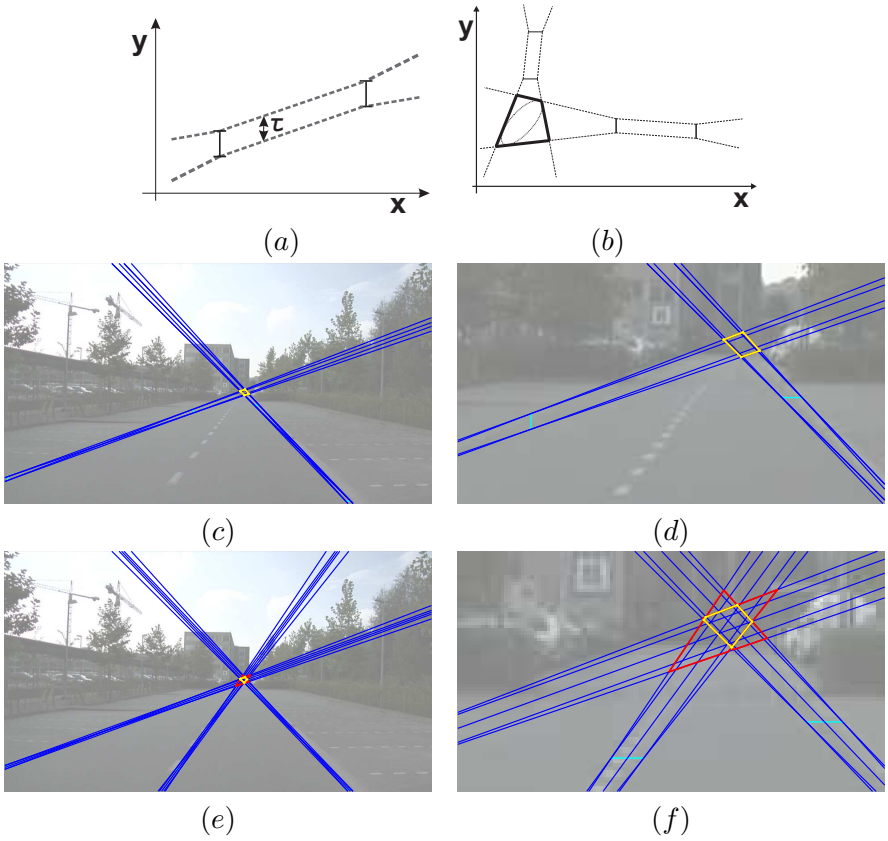


Figure 5.4: (a) A representation of the localization uncertainty of a straight line in the image domain. (b) The intersection of 2 uncertainty regions for straight lines is to be found in an uncertainty region of a rather irregular shape, that is not easy to capture by an ellipse. The vanishing point of 2 $(c-d)$ or more $(e-f)$ *parallel* world lines is well approximated by a polygonal uncertainty region.

i.e., the set of all Euclidean lines passing through 2 vertical segments of length τ. For lines with slope $|\alpha| > 1$, a similar definition can be given with 2 horizontal uncertainty segments. Figure 5.4 (b) shows that the uncertainty region can be represented as a bow-tie shaped region of a certain allowable thickness τ in the image domain.

The uncertainty on the intersection of both lines then is well represented by a polygonal region as shown in Figure 5.4 (b). The uncertainty region in which to look for the intersection of two uncertain lines is not at all shaped as an ellipse. Even more, there can be a consider-

able difference in volume of the enclosed and the enclosing ellipsoid of a convex set.

Figure 5.4 $(c-f)$ shows a practical example, where the intersection of two lines that are parallel in real world meet in a vanishing point in the image space. The uncertainty region in which to look for the intersection of the two uncertain localized lines is not at all shaped as an ellipse. This effect is even more visible when looking at the intersection region of more line uncertainty regions.

Another major limitation of statistical inference is that the computations soon become too complicated when propagating the uncertainty model throughout the computations of geometric instances, unless one introduces simplifying, but not always realistic assumptions about the statistical model. Using uncertainty polytopes often results in simple computations, and gives the possibility to proceed further in the geometric reasoning chain.

Figure 5.5 illustrates the propagation of uncertainty on the location of point features into higher-level geometric reasoning. Suppose we want to detect a circular shape in the image, then 3 points are required to locate the position of the circle center point (a). If the point localization uncertainty is represented as a square region in the image space, that uncertainty is propagated to the line parameters of the perpendicular bisectors. The bisectors pass through the midpoint of each line segment for which the endpoints are located in the uncertainty region (b). Similar as in Figure 5.4, an uncertainty region for the circle center is obtained as the convex hull of the set of intersection points of all possible bisectors (c). Figure 5.5 (d) shows a subset of circles from the parameter uncertainty polytope (i.e. the vertices) (e).

The uncertainty on the circle location can be propagated further in the geometric reasoning chain, e.g., in the computation of the tangent at the 3 initially chosen points. The uncertain location of the tangents in the image space is illustrated in Figure 5.5 (f). We can represent the uncertainty on the line parameters for the tangent as a convex region in the $\alpha\beta$ parameter space delimited by halfplanes (g).

In many correspondence problems it is often sufficient to have a reliable RoI for a feature, without the exact knowledge of a pdf for the feature locations. Consequently, our approach is widely applicable as it can not only cope with RoIs with a wide variety in shape, but can also be directly applied to transformations such as 2D affine and projective transformations, or perspective transformations.

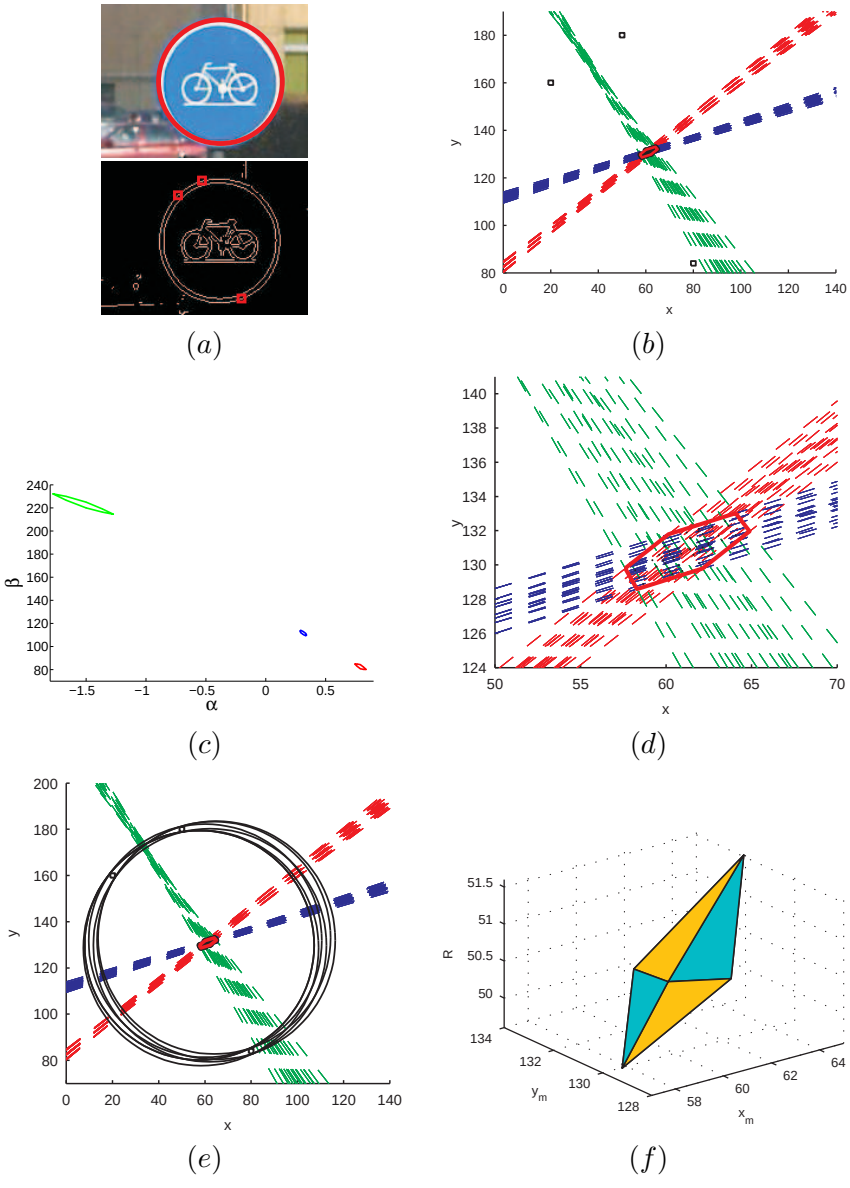
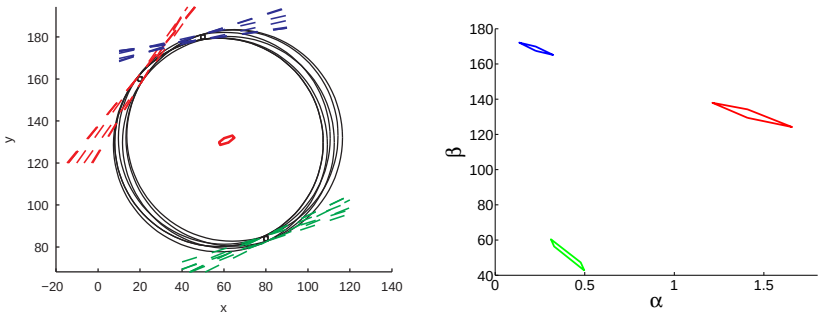


Figure 5.5: Figure continues on next page.



(g) (h)

Figure 5.5: An illustration of the propagation of uncertainty in a geometric reasoning chain in our framework. (a) The propagation of uncertainty on the parameters of a circle and its tangents from 3 locations considered uncertain. (b) The perpendicular bisectors of all line segments with endpoints at the vertices of the uncertainty regions. (c) The convex polygonal uncertainty regions for the bisectors in the $\alpha\beta$ parameter space (for a line equation of the form $y = \alpha x + \beta$). (d) The intersection of the line uncertainty regions results in the uncertainty region for the circle center point (x_m, y_m). (e) The circles corresponding to the vertices of the uncertainty polytope in the (x_m, y_m, R) parameter space (f). The uncertainty on the 3 tangents in the image (g) and the $\alpha\beta$ parameter (h) space.

When polytopes are used to capture uncertainty, many recognition problems in computer vision can be reformulated as either convex or combinatorial optimization problems for which standard algorithms are known. The proposed method is not seen as a replacement of statistical inference, but rather as an additional tool, in which we make use of the hard bounds often known for geometric parameters, such as minima and maxima for the intrinsic or extrinsic camera parameters, or the maximum velocity of objects. In fact, to get more precise information, the uncertainty ellipsoids can be embedded in the polytopes.

5.3 The Presented Uncertainty Framework

A matching process starts from two finite sets of features in two distinct but related image frames. Assume that the exact position of the features cannot be resolved due to errors introduced by the digitization process or by the feature detector. Furthermore, the features of one frame can be projected on the features of the second frame by a transformation map,

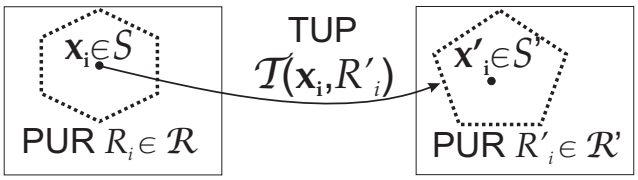


Figure 5.6: An illustration of the notations in our uncertainty framework.

but that the transformation map cannot be precisely determined. How can we then describe the uncertainty for geometric transformations?

We propose an uncertainty framework to model the uncertainty about feature locations and transformations by polygonal uncertainty regions or polytopes in parameter space. In the remainder of this chapter, we will introduce the mathematical notation for the uncertainty of transformations in greater detail.

We will assume from now on that all features in a sample set are in *general position*. Special configurations, such as collinear points or concurrent lines, were illustrated in the previous chapter. Here, we will not discuss these configurations in detail in the uncertainty framework as they do not pose additional difficulties, but would result in an abundance of additional technical details in the text.

5.3.1 Overview of Transformation Uncertainty Problems

In a correspondence problem we start from a set of detected features $\mathbf{x_i} \in S$ with a unique location in the first image and $\mathbf{x'_i} \in S'$ in the second image. However, as we have seen that the location returned by some feature detector can be uncertain, we model the localization uncertainty by defining a set of possible feature locations, the *positional uncertainty region* (PUR), in which the feature will occur, without knowing its exact position. Convex bounded polygons R_i of a certain size and shape are used to model the uncertainty about the exact location of a feature $\mathbf{x_i}$ in the image.

In our uncertainty framework, the collection of all PURs R_i will now be referred to as $\mathcal{R}$. Similarly, we define the collection $\mathcal{R}'$ of regions R'_i around the location of the features $\mathbf{x'_i} \in S'$ in the second image.

For a set of feature correspondence pairs $\{(\mathbf{x_i}, \mathbf{x'_i})\}$ we can compute a transformation T with unique parameters. When solving problems

with uncertain location parameters sets, the uncertainty of transformations is represented as a *transformation uncertainty polytope* (TUP) in n dimensions, one for each parameter of the transformation. The uncertainty for the transformation can be deduced naturally from the positional uncertainty, as we will show in the next section.

We use the notation $\mathcal{T}(\mathbf{x_i}, R'_i)$ to denote the TUP as the set of all transformations that each map the feature $\mathbf{x_i}$ onto a unique location in its corresponding uncertainty region R'_i (as illustrated in Figure 5.6). Then $\mathcal{T}(S, \mathcal{R}')$ denotes the set of transformations that map each feature $\mathbf{x_i}$ in the set S into the corresponding PURs R'_i in the collection $\mathcal{R}'$. To solve a RoI problem, we must determine the PUR R'_i into which the feature $\mathbf{x_i}$ is mapped given $\mathcal{T}$. We use the notation $R'_i(\mathbf{x_i}, \mathcal{T})$ to indicate the RoI problem, or $\mathcal{R}'(S, \mathcal{T})$ for a set S of features.

Both the computation of a TUP ($\mathcal{T}(S, \mathcal{R}')$) from a set of correspondences, and the computation of RoIs ($\mathcal{R}'(S, \mathcal{T})$) for a set of features are essential in a matching process. Therefore, we will introduce them in detail in sections 5.4.1 and 5.4.2 for point features.

$\mathsf{T}(S, \mathcal{R}')$ and $\mathsf{T}(\mathcal{R}, S')$ are always meaningless. For example, it is impossible to map set of a unique point by a single transformation onto all locations in a PUR, which excludes $\mathsf{T}(S, \mathcal{R}')$. Some problems may not have a solution in general. For example, given two arbitrary polygonal PURs, only when these two polygons are carefully chosen, the problem $\mathsf{T}(R_i, R'_i)$ or $\mathsf{T}(\mathcal{R}, \mathcal{R}')$ will have a solution. Likewise, we do not consider $S(\mathcal{R}', \mathcal{T})$ and $S'(\mathcal{R}, \mathcal{T})$.

Solving a RoI Problem

Suppose that an image can be mapped on a second image by a well-known transformation. This case occasionally occurs in practice.

For example, consider two observations of the same scene from two different camera positions, for which all extrinsic and intrinsic camera parameters are well-known, as illustrated in Figure 5.7. Suppose that the position of a line in the first image cannot be determined very precisely, but that we can define a bounded set of line parameters, describing its possible positions and angles. What are the possible line parameters in the second image? Or in other words, can we determine a RoI for the line in the second image? The solution is relatively straightforward as it suffices to compute $\mathcal{R}'(\mathcal{R}, \mathsf{T})$ to obtain the RoIs.

Solving a Correspondence Problem

Our goal is to present a robust matching procedure to solve correspondence problems when the transformation between images is unknown. We present a solution which essentially involves the backbone of a RANSAC process, with inclusion of our uncertainty framework. A solution for both the problem of computing a TUP ($\mathcal{T}(S, \mathcal{R}')$), and the RoI problem ($\mathcal{R}'(S, \mathcal{T})$) are essential herein. Figure 5.7 gives an illustration of this procedure, which we will term U-RANSAC from now on.

Since the camera parameters are generally unknown, several iterations of the following process are required to establish correct correspondences between features in both sets.

1. Find a set of geometric features in both images (points, lines, ...).
2. Determine a sample set P_s of features in the first image and their possible correspondences in the second image. Take the localization uncertainty into account as a set of PURs $R'_i \in \mathcal{R}'$. Square regions of a few pixels high often suffice to capture the uncertainty introduced by the feature extraction process (see section 2.4). A careful sample selection can considerably reduce the number of required iterations (see 3.3.1).
3. Compute a TUP $\mathcal{T}(S, \mathcal{R}')$ for the set of possible transformations. Often prior information regarding the transformation can be incorporated in the computations.
4. Use $\mathcal{T}$ to compute RoIs for all the other features $\mathbf{x_j} \in S_o$ not in S, that is, solve for $\mathcal{R}'_o(S_o, \mathcal{T})$. The size and shape of the TUP determine the variation of the size and the shape of the RoIs across the image. Use the RoIs to find the actual correspondences.

A first important difference to the standard framework is that we do not consider one estimate for the transformation model, but all models within a polytope in parameter space. This gives the advantage of not rejecting good, but inaccurate samples. The correspondence set for the TUP computed with corrupted or inaccurate samples will contain more true inliers than in the standard procedure.

A second advantage of our U-RANSAC framework is the natural approach of incorporating a priori known information about the transformation in the matching process, which is almost always available in practical applications.

The next chapter will consider the U-RANSAC process in more detail. The remainder of this chapter will introduce our solution for the

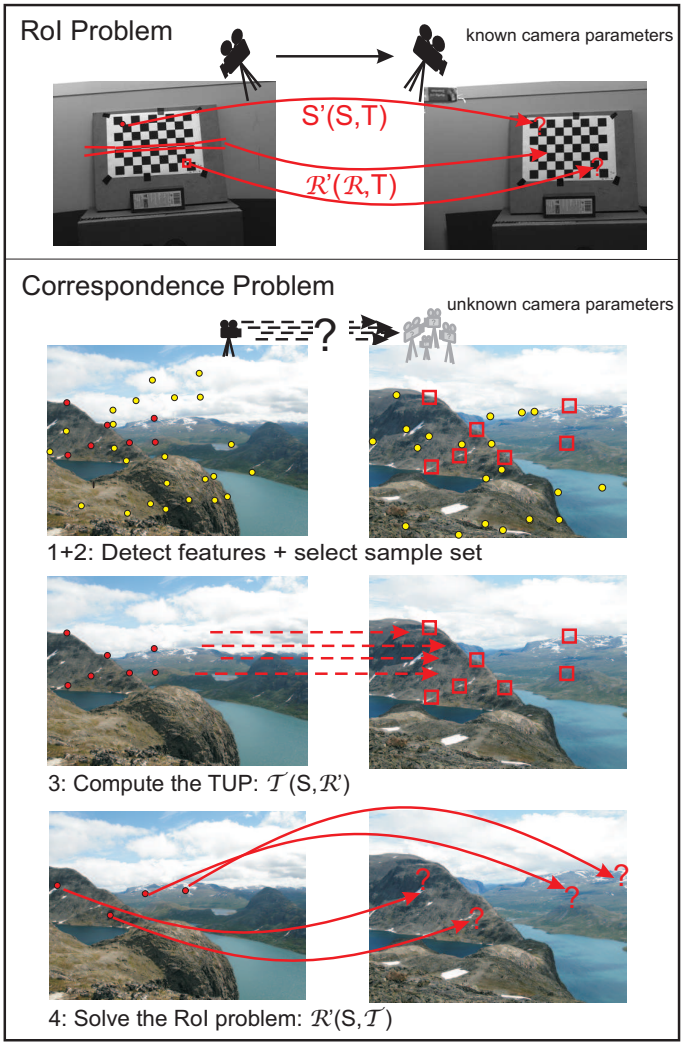


Figure 5.7: Our techniques to solve transformation problems can be applied to compute correspondences when the transformation parameters are either known or not. For known parameters, the solution for the RoI problem is straightforward. The other case is more challenging, as it requires a robust estimation approach (like U-RANSAC) to solve the correspondence problem.

computations of the TUPs and the RoIs. We will discuss our uncertainty approach for transformations of respectively points and lines in sections 5.4 and 5.5.

5.4 Transformation Uncertainty for Point Features

In this section, we describe the basics of the mathematical uncertainty framework and its use in the different transformation problems for point features. Both $S'(S, \mathsf{T})$ and $S(S', \mathsf{T})$ are trivial problems, which are solved by simply applying the forward transformation $\mathbf{x}' = \mathsf{A}\mathbf{x}$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.2}$$

or its inverse.

The uncertainty of the transformation map is determined by solving $\mathcal{T}(S, \mathcal{R}')$, as section 5.4.1 will describe. Then section 5.4.2 shows how to compute RoIs by using the uncertainty of the map as $\mathcal{R}'(S, \mathcal{T})$.

The cases $\mathcal{T}(S, \mathcal{R}')$ and $\mathcal{T}(S', \mathcal{R})$ are dual to each other, and so are $\mathcal{R}'(S, \mathcal{T})$ and $\mathcal{R}(S', \mathcal{T})$. $\mathcal{T}(S, \mathcal{R}')$ leads to the construction of TUPs for image features, while for problems that fall under $\mathcal{T}(S', \mathcal{R})$ it is easier to derive such polytopes or polygons for inverse transformations. Both cases will be discussed in section 5.4.3 for point features. $\mathcal{T}(\mathcal{R}, \mathcal{R}')$ is the most general case, but invariably leads to non-linear programming problems, as we will illustrate for point features in section 5.4.5.

5.4.1 Uncertainty of Transformations

The uncertainty of a transformation is described by allowing the transformation parameters to vary within a certain range. We describe the set of possible transformations as a convex polyhedron in n dimensions (one dimension for each transformation parameter).

From now on, we shall assume that the transformation polyhedron is always convex and bounded, i.e., that the polyhedron is a polytope [Ziegler, 1995]. This can be accomplished either by *providing a sufficient number of point pairs in general position*, i.e., 3 or more for the affine transformation, or by *imposing realistic constraints* on the transformation parameters, which are almost always available in an application. The TUP can then be represented as a set of linear inequalities in the transformation parameter space.

First, a TUP can be obtained in the parameter space by posing a sufficient number of constraints that are known beforehand. Consider

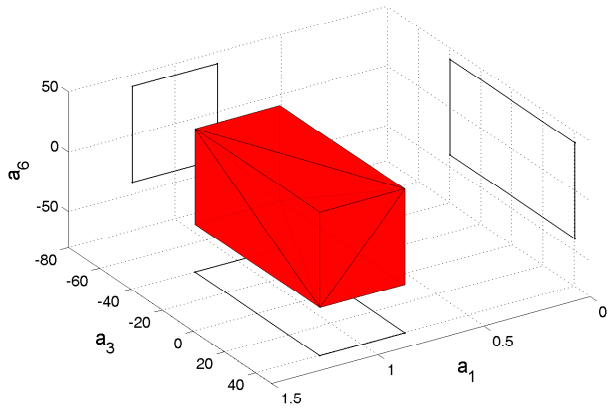


Figure 5.8: A TUP can be obtained by imposing constraints on the parameters. As an example, the shown polytope is obtained by constraining the parameters of a transformation of the form (5.3) as $-40 \leq a_3, a_6 \leq +40, 0.8 \leq a_1 \leq 1.2$.

an example for a transformation of the form

$$\left\{ \begin{array}{rcl} x' & = & a_1 x + a_3 \\ y' & = & a_1 y + a_6 \end{array} \right. , \tag{5.3}$$

i.e., translation and isotropic scaling. If the expected translation is maximum 40 pixels, then $-40 \leq a_3, a_6 \leq +40$, and the scaling is known not to exceed a factor 0.2, then we obtain the TUP in the 3 dimensional parameter space shown in Figure 5.8. Another option is to demand that the parameters cannot become very large, e.g., $|a_3| \leq 1000$.

Similarly, we can construct a TUP for a given correspondence set. To model the uncertainty of the image $\mathbf{x}' = (x', y')$ of a point $\mathbf{x} = (x, y)$ in $\mathbb{R}^2$, we assume that the PUR R' for $\mathbf{x}'$ is given as a convex polygon bounded by n halfplanes,

$$r_i x' + s_i y' \geq 1, \quad 1 \leq i \leq n \tag{5.4}$$

in the $x'y'$-plane.

We must now find an adequate description for the uncertainty on the transformation parameters derived from the positional uncertainty.

Definition 5.1 *Let* $\mathbf{x}$ *be a point in* $\mathbb{R}^2$*, and let be* R' *be a convex subset of* $\mathbb{R}^2$*. Then* $\mathcal{T}(\mathbf{x}, R')$ *denotes the set of all transformations* T *that map* $\mathbf{x}$ *into the PUR* R'*.*

$\mathcal{T}(\mathbf{x}, R')$ is a convex polyhedron in 6 dimensions supported by the hyperplanes that can be found by substituting the equations for the possible image of $\mathbf{x}$ under any transformation $\mathsf{T} \in \mathcal{T}(\mathbf{x}, R')$

$$\begin{cases} x' &= a_1 x + a_2 y + a_3 \\ y' &= a_4 x + a_5 y + a_6 \end{cases} \tag{5.5}$$

in (5.4), yielding the inequalities

$$r_i(a_1 x + a_2 y + a_3) + s_i(a_3 x + a_4 y + a_6) \geq 1, \quad 1 \leq i \leq n \tag{5.6}$$

for the parameters $a_1, \ldots, a_6$.

Similarly, if S is a finite set of points $\mathbf{x_i} \in \mathbb{R}^2$, $\mathcal{R}'$ a collection of corresponding PURs $R'_i \subset \mathbb{R}^2$, we now consider the problem of finding the set $\mathcal{T}(S, \mathcal{R}')$ of affine transformations which map each $\mathbf{x_i} \in S, i = 1, \ldots, n$ onto $R'_i \in \mathcal{R}'$ respectively. Then $\mathcal{T}(S, \mathcal{R}') = \cap_i \mathcal{T}(\mathbf{x_i}, R'_i)$.

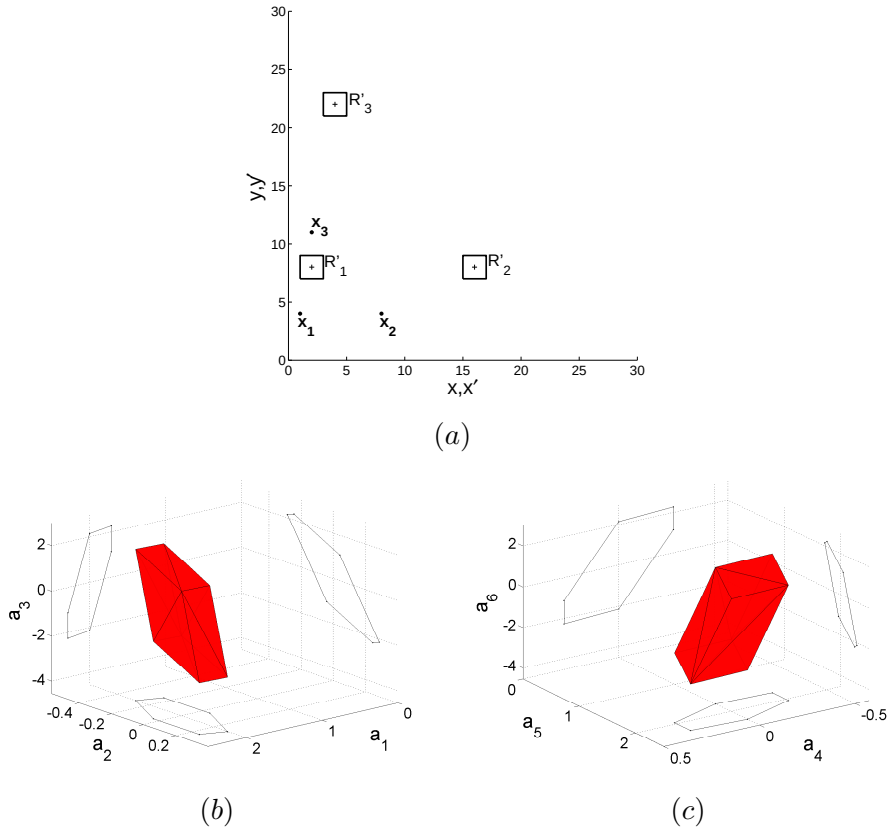
When S contains enough points, i.e., 3 for the affine transformation, the polyhedron $\mathcal{T}(S, \mathcal{R}')$ is bounded. Then $\mathcal{T}(S, \mathcal{R}')$ is a convex polytope of transformations in a 6-dimensional space, because the intersection of polytopes is again a polytope. An example is illustrated in Figure 5.9.

Properties. The transformation uncertainty is decreased by the *addition of more (correct) correspondence pairs*, or when the *uncertainty about the localization of the features is smaller*. Both observations are illustrated respectively in Figure 5.10 and 5.11.

The more certain one is about the localization of the features, the less uncertainty should remain about the transformation parameters. When the size of a PUR R' decreases, then also the size of the TUP $\mathcal{T}(\mathbf{x}, R')$ decreases. In fact, suppose that $R'_1 \subseteq R'_2$, then $\mathcal{T}(\mathbf{x}, R'_1) \subseteq \mathcal{T}(\mathbf{x}, R'_2)$.

A similar effect is obtained by the addition of extra point-region correspondence pairs to S and $\mathcal{R}'$. Then the size of the TUP $\mathcal{T}(S, \mathcal{R}')$ can only decrease, because of the additional set of halfplanes. Thus, assuming $\mathcal{R}'_1 \subseteq \mathcal{R}'_2$ and $S_1 \subseteq S_2$, then $\mathcal{T}(S_2, \mathcal{R}'_2) \subseteq \mathcal{T}(S_1, \mathcal{R}'_1)$.

Dual representation. Next to the representation of polytopes by inequalities, we can also consider a dual representation by the vertices of the polytope. A polytope can then be considered as the closed, convex span of its finite set of vertices. This assumption is supported by the



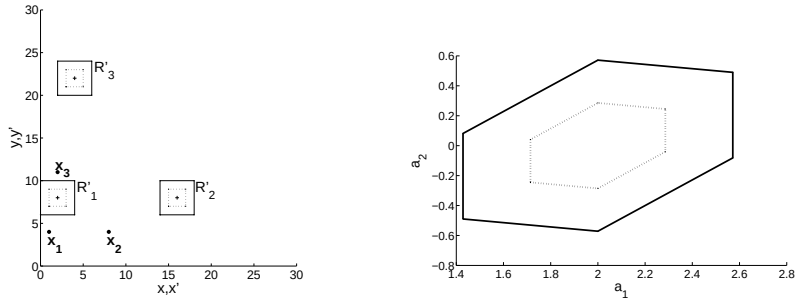
(a) (b) (c)

Figure 5.9: A TUP can be obtained for a sufficiently large set of correspondences in general position, e.g., 3 for the affine transformation. When defining the PURs R'_i in $\mathcal{R}'$ as squares of height 2 centered on $\mathbf{x}'$, we obtain a TUP, which is shown here as a projection in the 3 dimensional subspaces $a_1a_2a_3$ and $a_4a_5a_6$.

following two simple, but essential properties for convex combinations of transformations.

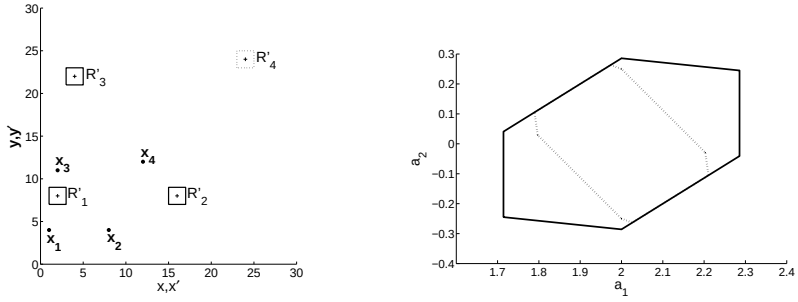
Proposition 5.2 *An affine transformation of a convex combination of points is equal to the convex combination of the transformed points, i.e.,* $\mathsf{T}(\alpha_1\mathbf{x_1} + \ldots + \alpha_n\mathbf{x_n}) = \alpha_1\mathsf{T}(\mathbf{x_1}) + \ldots + \alpha_n\mathsf{T}(\mathbf{x_n})$*, with* $\alpha_1 + \ldots + \alpha_n = 1$ *and* $0 \leq \alpha_i \leq 1$.

Proposition 5.3 *The application of a convex combination of affine transformations to a point* $\mathbf{x}$ *is equal to the convex combination of that point under the*



(a) (b)

Figure 5.10: When the features are more accurately localized, the uncertainty on the transformation also decreases. For different sizes of PURs R_i (dashed vs. solid lines (a)), the corresponding TUPs are given (dashed vs. solid lines (b)) (projected onto the a_1a_2-plane).



(a) (b)

Figure 5.11: A TUP gets smaller as a result of the addition of correspondences. (a) Recompute the TUP after the addition of a correspondence pair $(\mathbf{x_4}, R'_4)$ to the set $\{(\mathbf{x_i}, R'_i)\}, i = 1 \ldots 3$. (b) The TUP is larger in the case of 3 correspondences (solid line), than in the case of 4 correspondences (dashed line) (projected onto the a_1a_2-plane).

different transformations, i.e., $(\alpha_1\mathsf{T}_1 + \ldots + \alpha_n\mathsf{T}_n)(\mathbf{x}) = \alpha_1\mathsf{T}_1(\mathbf{x}) + \ldots + \alpha_n\mathsf{T}_n(\mathbf{x})$*, with* $\alpha_1 + \ldots + \alpha_n = 1$ *and* $0 \leq \alpha_i \leq 1$.

More details are given in [Teelen and Veelaert, 2009].

5.4.2 Implied Uncertainty Regions

Given the TUP and a set of feature points, we can solve the RoI problem. The solution follows directly from Proposition 5.3. The RoI is the

uncertainty region $R'(\mathbf{x}, \mathcal{T})$ *implied* by all transformations in $\mathcal{T}$.

For example, suppose that $\mathcal{T}$ is given as a polytope bounded by halfspaces in the parameter space $a_1, \ldots, a_6$:

$$\begin{cases} h_1a_1 + i_1a_2 + j_1a_3 + k_1a_4 + l_1a_5 + m_1a_6 & \leq & 1 \\ h_2a_1 + i_2a_2 + j_2a_3 + k_2a_4 + l_2a_5 + m_2a_6 & \leq & 1 \\ \ldots & & \\ h_na_1 + i_na_2 + j_na_3 + k_na_4 + l_na_5 + m_na_6 & \leq & 1. \end{cases} \tag{5.7}$$

Then $R'(\mathbf{x}, \mathcal{T})$ is the set of all points $\mathbf{x}' = (x', y')$ satisfying

$$\begin{cases} x' & = a_1x + a_2y + a_3 \\ y' & = a_4x + a_5y + a_6 \end{cases} \tag{5.8}$$

where $a_1, \ldots, a_6$ are solutions of (5.7).

To be precise, let $\mathbf{x}$ be a point and $\mathcal{T}$ a given TUP, then $R'(\mathbf{x}, \mathcal{T})$ denotes the *implied uncertainty region* (IUR) resulting from mapping $\mathbf{x}$ by all transformations in $\mathcal{T}$.

Definition 5.4 *Let $\mathbf{x}$ be a point in $\mathbb{R}^2$, and let $\mathcal{T}$ be a given transformation polytope. Then $R'(\mathbf{x}, \mathcal{T})$ denotes the IUR that results from mapping $\mathbf{x}$ by transformations in $\mathcal{T}$, i.e., $R'(\mathbf{x}, \mathcal{T}) = \{\mathbf{x}' \in \mathbb{R}^2 : \mathbf{x}' = \mathsf{T}(\mathbf{x})$ for some $\mathsf{T} \in \mathcal{T}\}$.*

An example is given in Figure 5.12, where a set of points is mapped by the set of transformations that are enclosed by the TUP of the example given in Figure 5.9. The points mapped by these transformations lie in an IUR, the boundaries of which can be determined accurately by the following property.

Proposition 5.5 *Let $\mathbf{x}$ be a point in $\mathbb{R}^2$, and let $\mathcal{T}$ be a given TUP. Let V_i denote the transformations that correspond to the vertices of the polytope $\mathcal{T}$. Then $R'(\mathbf{x}, \mathcal{T})$ is the convex hull of the points $\mathbf{x}'_\mathbf{i}$, where $\mathbf{x}'_\mathbf{i} = \mathsf{V}_i(\mathbf{x})$.*

The IURs for the points in S are contained in the PURs in $\mathcal{R}'$ for which the polytope $\mathcal{T}(S, \mathcal{R}')$ is computed.

Proposition 5.6 *Let S be a finite set of points $\mathbf{x_i}$, $\mathcal{R}'$ the collection of the corresponding PURs R'_i. Furthermore, let $\mathcal{T}$ be the TUP determined by the points $\mathbf{x_i}$ and their corresponding regions R'_i, that is $\mathcal{T} = \mathcal{T}(S, \mathcal{R}')$. Then $R'(\mathbf{x_i}, \mathcal{T}) \subseteq R'_i$ for each $\mathbf{x_i} \in S$.*

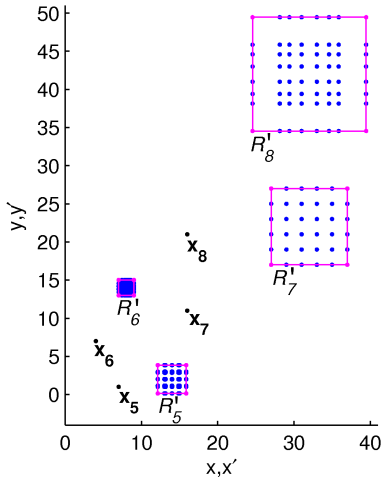


Figure 5.12: The IUR $R'(\mathbf{x}, \mathcal{T})$ for $\mathbf{x_i}$ is the set of points that are found as a mapping of $\mathbf{x_i}$ by the transformations in $\mathcal{T}$. The IURs $R'(\mathbf{x}, \mathcal{T})$ are the convex hull of the points $\mathsf{V}_i(\mathbf{x})$ where the transformations V_i denote the vertices of the polytope $\mathcal{T}$ (Proposition 5.5). In this example, $\mathcal{T}$ is the TUP obtained for the situation illustrated in Figure 5.9.

This property is illustrated in Figure 5.13. The TUP is computed for the set of correspondence pairs $\{(\mathbf{x_i}, R'_i)\}$ with the points $\mathbf{x_i} \in S$ and the uncertainty polygons $R'_i \in \mathcal{R}'$ (a). The IURs $R'(\mathbf{x_i}, \mathcal{T})$ are subsets of the regions R'_i (b).

Shape, size and position of implied regions. One of the advantages of our framework is that it can cope with a wide variety in shape of the uncertainty polygons. This is illustrated by a few examples which show how shape and size of the IURs varies in function of the sample subset of correspondences (and thus the computed TUP).

Shape. For the example presented in Figure 5.14, the point set $S = \{\mathbf{x_1}, \ldots, \mathbf{x_4}\}$ and the collection of PURs $\mathcal{R}' = \{R'_1, \ldots, R'_4\}$ (triangular or square) determine a TUP $\mathcal{T}(S, \mathcal{R}')$. Then an arbitrary point $\mathbf{x} = (x, y)$ will be mapped to an IUR $R'(\mathbf{x}, \mathcal{T})$ that is either triangular, or square, or some intermediate shape. We see that the IURs $R'(\mathbf{x_5}, \mathcal{T}(S, \mathcal{R}'))$ and $R'(\mathbf{x_6}, \mathcal{T}(S, \mathcal{R}'))$ have a shape between triangular and square.

This effect can be explained as follows. Consider the 8-dimensional

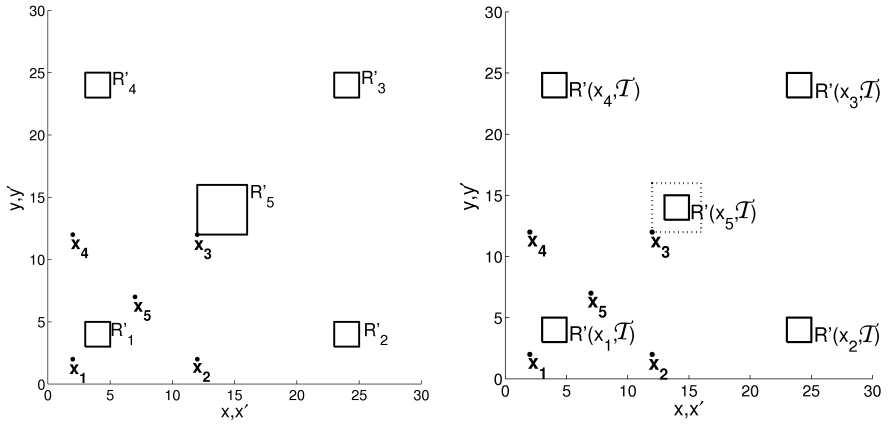


Figure 5.13: (a) The TUP $\mathcal{T}$ is computed for the 5 correspondence pairs $(\mathbf{x_i}, R'_i)$ with the points $\mathbf{x_i} \in S$ and the uncertainty polygons $R'_i \in \mathcal{R}'$. (b) The IURs $R'(\mathbf{x_i}, \mathcal{T})$ (solid lines) are subsets of the regions R'_i (dashed lines). Actually, for $i = 1 \ldots 4$, even $R'(\mathbf{x_i}, \mathcal{T}) = R'_i$.

space formed by the 6 parameters $a_1, \ldots, a_6$ of an affine transformation and the coordinates (x', y') of an image point. The points $\mathbf{x_1}, \ldots, \mathbf{x_4}$ and the inequalities of the PURs $R'_1, \ldots, R'_4$ define a polyhedron of the form (5.7) in this 8-dimensional space. Consider the intersection polytope of this polyhedron with the linear subspace defined by (5.8), in which we may substitute, for example, the coordinates (x, y) of the point $\mathbf{x_5}$. Then the region $R'(\mathbf{x_5}, \mathcal{T})$ is the projection of this polytope on the $x'y'$-plane. In fact, by substituting different coordinates (x, y) in (5.8), we get different intersections of the polyhedron (5.7), and therefore different regions. In the case of Figure 5.14, the shape of an IUR $R'(\mathbf{x}, \mathcal{T})$ is triangular, square, or some intermediate shape, depending how the linear space (5.8) intersects the polyhedron (5.7).

Assuming that a convex polygon can accurately approximate the shape of most PURs, even if defined in other ways (e.g. by thresholding a pdf), our uncertainty framework can cope with an arbitrary shaped PUR.

Size. The variation in size of IURs depends on their position in the image space. Figure 5.15 shows an example where 3 points $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ and their corresponding regions R'_1, R'_2, R'_3 restrict the set of possible

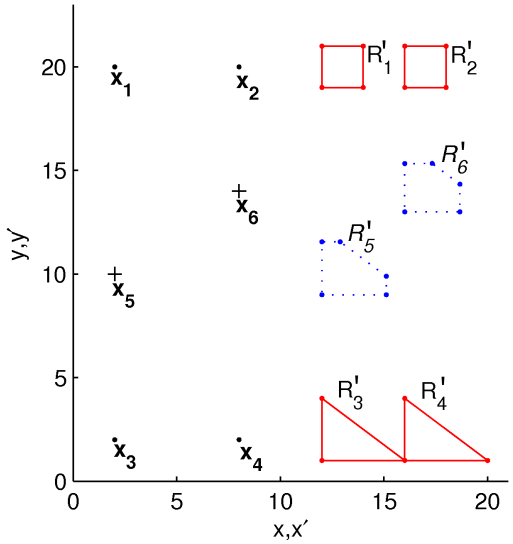


Figure 5.14: The set of points $\mathbf{x_i} \in S, i = 1, \ldots, 4$ and their corresponding PURs $R'_i \in \mathcal{R}'$ determine $\mathcal{T}(S, \mathcal{R}')$. This TUP implies the regions $R'(\mathbf{x_5}, \mathcal{T}(S, \mathcal{R}'))$ and $R'(\mathbf{x_6}, \mathcal{T}(S, \mathcal{R}'))$.

transformations to the TUP $\mathcal{T} = \cap_{i=1,\ldots,3}\mathcal{T}(\mathbf{x_i}, R'_i)$. With $\mathcal{T}$, we can compute the PUR $R(\mathbf{x_j}, \mathcal{T})$ for any point $\mathbf{x_j}$ in the plane. Some of these regions are shown in Figure 5.15.

Notice that the size of the regions grows linearly, when moving away from the initial correspondences. In the local neighborhood of the initial correspondences, the uncertainty about the location of the transformed features is rather small. When moving farther away from the sample set, the uncertainty grows.

We can accurately determine the size of the IURs based on the position of each $\mathbf{x}$ in the image, without actually computing the transformation of $\mathbf{x}$ under all vertices of the TUP. In fact, for transformations of the form

$$\begin{cases} x' &= a_1 x + a_3 \\ y' &= a_5 y + a_6 \end{cases},$$

and rectangular PURs, we can prove that the varying size of rectangular IURs must increase piecewise linearly, according to either the value of the x or y coordinate.

We know that the IUR for a point $\mathbf{x}$ is the convex hull of the points

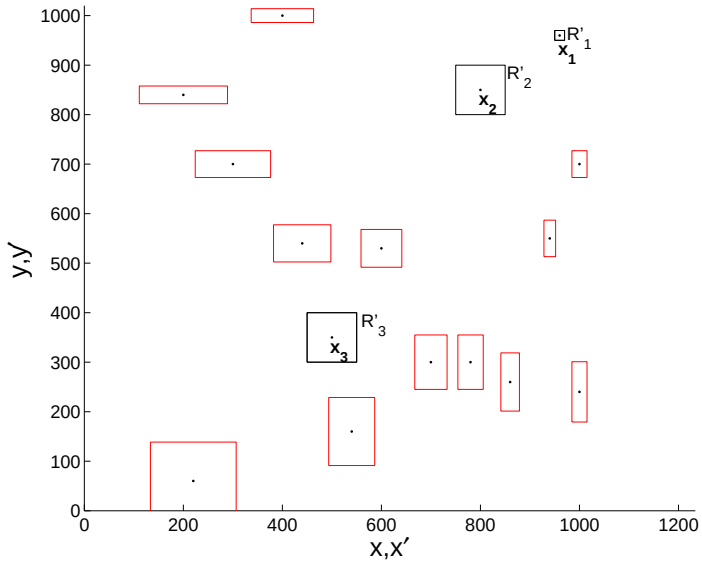


Figure 5.15: The TUP is determined by $(\mathbf{x_i}, R'_i), i = 1, \ldots, 3$. Notice how the size of the IURs $R'(\mathbf{x}, \mathcal{T})$ for a number of unnamed points varies with the position of the unnamed points.

$\mathsf{V}_i(\mathbf{x})$, where the V_i are the vertices of the TUP (Proposition 5.5). Suppose we have a TUP with vertices $(a_{i,1}, a_{i,3}, a_{i,5}, a_{i,6})$. Then the IURs have vertices $(a_{i,1}x + a_{i,3}, a_{i,5}y + a_{i,6})$. Hence the width of the IURs varies along the x-direction as the upper bound of all $|(a_{i,1}x + a_{i,3}) - (a_{j,1}x + a_{j,3})|$. Likewise, for the y-direction the height varies as the upper bound on $|(a_{i,5}y + a_{i,6}) - (a_{j,5}y + a_{j,6})|$.

In Figure 5.16 (a), we choose 3 correspondences $(\mathbf{x_i}, R'_i)$ to compute the TUP. For all other indicated points we compute the IURs with that TUP. Notice that the IURs in between the three points $\mathbf{x_i}$ are of the same size. The size of the IURs for other unnamed points grows piecewise linearly away from the $\mathbf{x_i}$. Figure 5.16 $(b - c)$ shows how the size of the IURs varies as a function that has the shape of a *bow tie*. The bow tie shape is similar to the shape obtained for the collinearity of digital straight lines or the uncertainty cones for point-line coincidences [Veelaert, 2001; Förstner, 2004]. Similar properties can be defined for an affine transformation of the form (5.2).

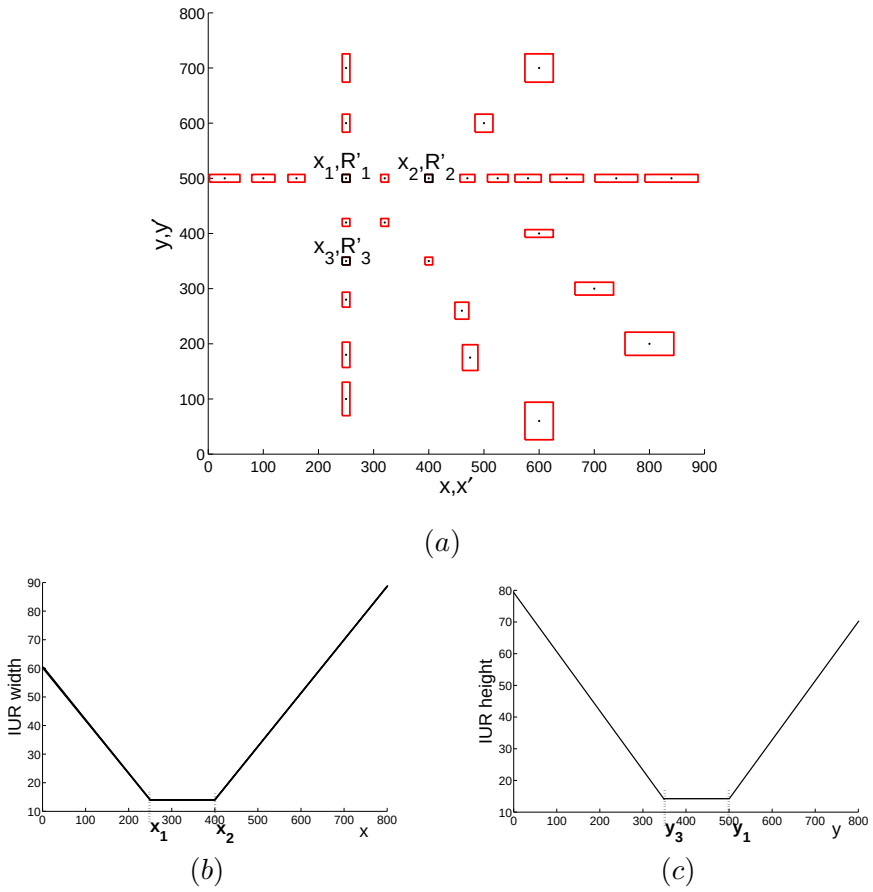


Figure 5.16: The set S of points $\mathbf{x_i}, i = 1, \ldots, 3$ and their corresponding PURs $R'_i \in \mathcal{R}'$ determine $\mathcal{T}(S, \mathcal{R}')$. With $\mathcal{T}(S, \mathcal{R}')$ the IURs for all other (unnamed) points are computed. The size of the IURs varies as the shape of a bow tie in horizontal (b) and vertical direction (c).

Practical example. In Figure 5.17, we present a simple example of how to compute the RoIs for an affine TUP. If a sample subset of correspondences $\{(\mathbf{x_i}, R'_i)\}$ is known or randomly selected $(a - b)$, then we can compute the TUP $\mathcal{T}$. The RoIs for a larger set S of detected Harris features in (c) can then be computed as the IURs $\mathcal{R}'(S, \mathcal{T})$. The resulting regions are shown in (d).

Notice how the size of the RoIs varies over the image, when the distance to the features in the sample set increases. The height of the IURs remains constant within the convex hull of the 3 $\mathbf{x}'_i$ in the sample

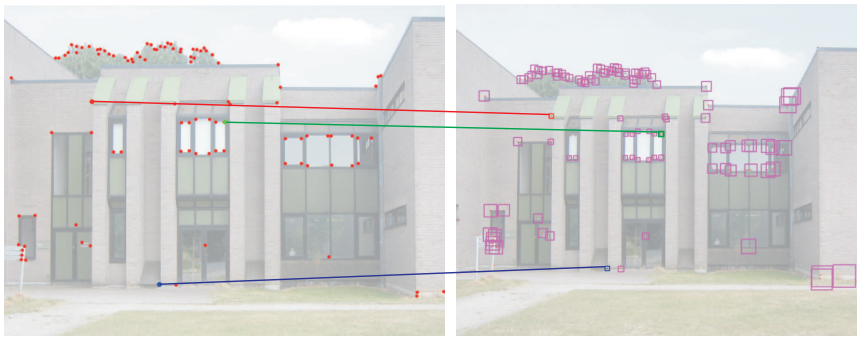


Figure 5.17: An example of the computation of RoIs for affine TUPs. The TUP is computed from the indicated sample set of 3 correspondences. Then the RoIs for a set of Harris features in the left image are obtained as their IURs in the right image.

set. We can say that the uncertainty there is less, and increases when moving away from the initial set.

5.4.3 Transformation Uncertainty Polytope Duality

Suppose we have a TUP of affine transformations, what is the shape of the set of inverse transformations? It is *not* a polytope, and not even a convex set.

Note that the inverse of an affine transformation is also an affine transformation. For an affine transformation A of the form (5.2), the inverse transformation A^{-1} is given by

$$\mathsf{A}^{-1} = \begin{bmatrix} \frac{a_5}{-a_2a_4+a_1a_5} & \frac{-a_2}{-a_2a_4+a_1a_5} & \frac{-a_5a_3+a_2a_6}{-a_2a_4+a_1a_5} \\ \frac{-a_4}{-a_2a_4+a_1a_5} & \frac{a_1}{-a_2a_4+a_1a_5} & \frac{a_4a_3-a_1a_6}{-a_2a_4+a_1a_5} \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.9}$$

We will give an example to illustrate that the inverse of a TUP is not a polytope. We choose a TUP composed of a convex combination of transformations $\alpha_1\mathsf{T}_1 + \ldots + \alpha_3\mathsf{T}_3, 0 \leq \alpha_i \leq 1$ and $\sum \alpha_i = 1$ (Figure 5.18 (c)). This TUP maps the points $\mathbf{x_i}$ in (a) onto the line segments in (b). A projection of the calculated set of inverse transformations on the parameter plane a_1a_2 is visualized in Figure 5.18(d). The projection is not convex, thus also the set of inverse transformations is not a convex polytope.

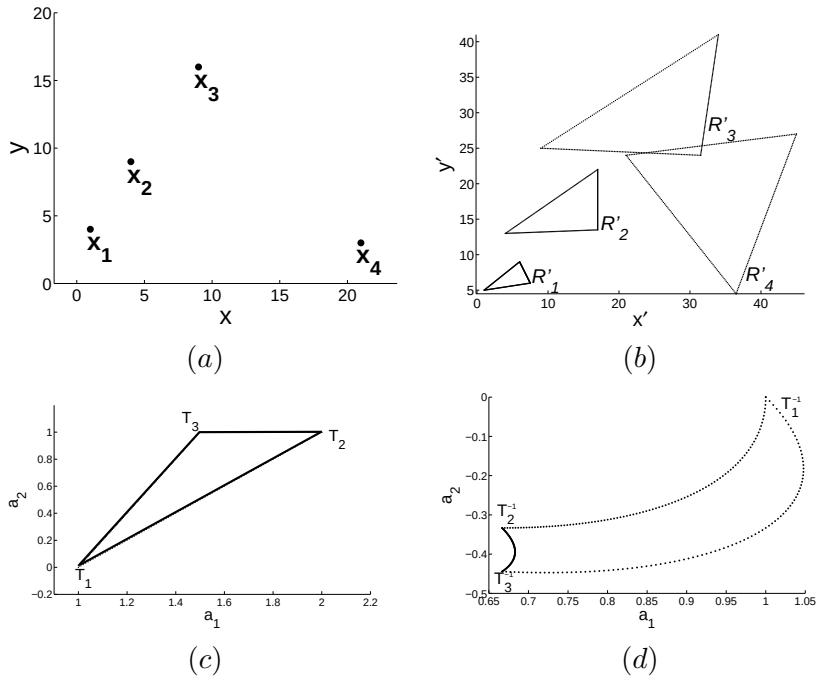
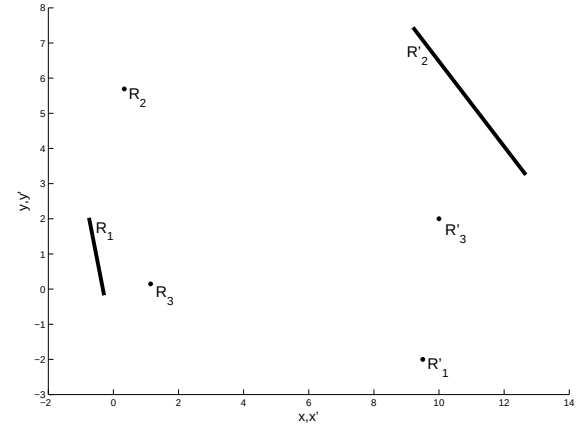


Figure 5.18: We consider a TUP as the convex combination of $\mathsf{T}_1 = [1,0,0;1,1,0;0,0,1]$, $\mathsf{T}_2 = [2,1,0;1,2,0;0,0,1]$, and $\mathsf{T}_3 = [1.5,1,2;0,1.5,0;0,0,1]$. (c) A projection of the TUP onto the a_1a_2 parameter plane. This TUP maps each point $\mathbf{x_i}$ in (a) to its corresponding PUR R'_i (b). (d) The projection of the set of inverse transformations on the a_1a_2 parameter plane.

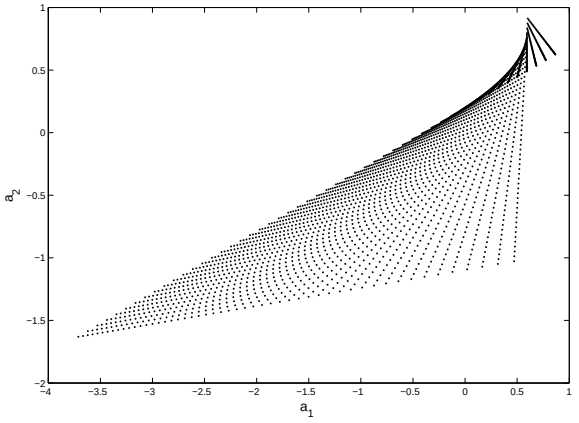
Even if we restrict the transformations to scaling and translation, the set of inverse transformations is not convex. Counterexamples can easily be found for those cases. Only if the transformations are restricted to translations, the inverse transformations form a polytope.

5.4.4 Known Transformation Parameters

When the transformation parameters are known but the location of a feature is uncertain, we need to determine the collection of regions $\mathcal{R}'(\mathcal{R},\mathsf{T})$ to determine the possible locations of the transformed feature in another image. It is straightforward to compute $\mathcal{R}'(\mathcal{R},\mathsf{T})$, based on Proposition 5.2. When the $\mathbf{x_i}$ denote the vertices of a polygonal PUR in the source plane, the PUR in the image plane is the convex hull of the



(a)



(b)

Figure 5.19: (a) The PURs in the source (R_i) and image (R_i') plane (for a special case where R_1 and R_2' degenerate to line segments and the other R_i to points). (b) The projection of the transformation uncertainty set from the R_i into the R_i' on the a_1a_2 parameter plane.

points $\mathsf{T}(\mathbf{x_i})$.

In this case, the inverse problem $\mathcal{R}(\mathcal{R}', \mathsf{T})$ can also be solved. Simply apply the inverse transformation to the vertices $\mathbf{x_i'}$ of the uncertainty polygon in the image plane. Then the PUR in the source plane is the convex hull of the points $\mathsf{T}^{-1}(\mathbf{x_i'})$.

5.4.5 The Most General Problem

The most general problem, when the point locations in both the source and destination image, and the transformation parameters are uncertain, invariably leads to non-linear equations.

Let R_1, R_2, R_3 and R'_1, R'_2, R'_3 be convex PURs in the source and the image plane, respectively. Then the set of affine transformations that map at least one point of R_i into R'_i is not convex. The union of all polytopes $\mathcal{T}(\mathbf{x_i}, R'_i)$ that can be computed for each $\mathbf{x_i} \in R_i$ is not necessarily a polytope.

We can illustrate this by a simple example. Figure 5.19 (a) shows the uncertainty polygons R_1, R_2, R_3 and their images R'_1, R'_2, R'_3 (for a special case where R_1 and R'_2 degenerate to line segments and the other R_i, R'_i to points). Figure 5.19 (b) shows the projection of a regular sampling of affine transformations in the uncertainty set onto the a_1a_2 parameter plane. It clearly shows that the projected set is not convex, which means that also the transformation uncertainty set is not convex, and certainly not a polytope in this case.

5.5 Transformation Uncertainty for Straight Lines

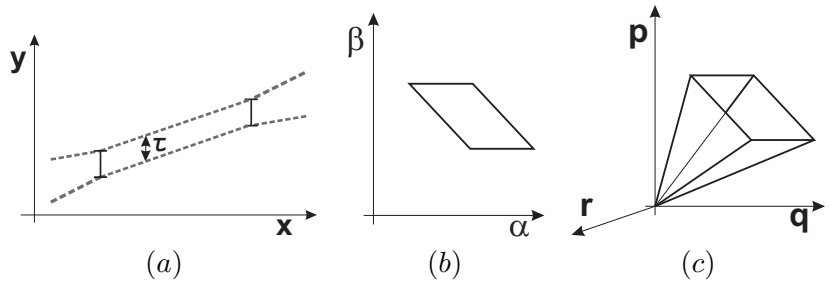
This section considers the transformation problems introduced above, but now for straight lines instead of points. The edge map of many imaged scenes can be segmented into straight lines, e.g., a lot of manmade structures are objects with straight boundaries. The image processing approaches that segment and extract digital straight lines are precise up to some usually predefined bound. This means that all pixels of which the line is composed lie within a certain distance of the location specified by the line equation. Therefore it makes sense to define a PUR for each detected line.

5.5.1 Representing Uncertainty for a Straight Line

If the equation of a straight line is written as $px+qy+r = 0$, its positional uncertainty region (PUR) in the image space is given as

$$-\frac{\tau}{2} \leq y_i - (-\frac{p}{q}x_i - \frac{r}{q}) \leq \frac{\tau}{2},\ i = 1, 2. \tag{5.10}$$

The region is composed of all lines that pass through two intervals of height τ at the points $\mathbf{x_1}$ and $\mathbf{x_2}$ for lines with slope $\alpha = -\frac{p}{q} \leq 1$ and



(a) (b) (c)

Figure 5.20: The localization uncertainty of a line can be represented in the xy image space (a), the $\alpha\beta$ (b) and pqr parameter domain (c).

arbitrary intercept $\beta = -\frac{r}{q}$. A similar definition with 2 horizontal segments can be given for lines with slope > 1.

Figure 5.20 (a) shows this PUR as a bow tie shaped region of a certain thickness in the image domain. The localization uncertainty can be represented as a convex region in the $\alpha\beta$ parameter space (b).

Any point on the ray $\gamma(p, q, r) = (\gamma p, \gamma q, \gamma r)$, with $\gamma \neq 0$, represents the same line. In the pqr parameter domain, the uncertainty for line parameters can then be represented by a pyramid C minus the origin, where C contains a set of rays of the form $\gamma_i(p_i, q_i, r_i)$. Figure 5.20 (c) shows an example of the uncertainty pyramid (PUP) in the parameter space. This is also termed the pre-image of a discrete line [Veelaert, 1999b].

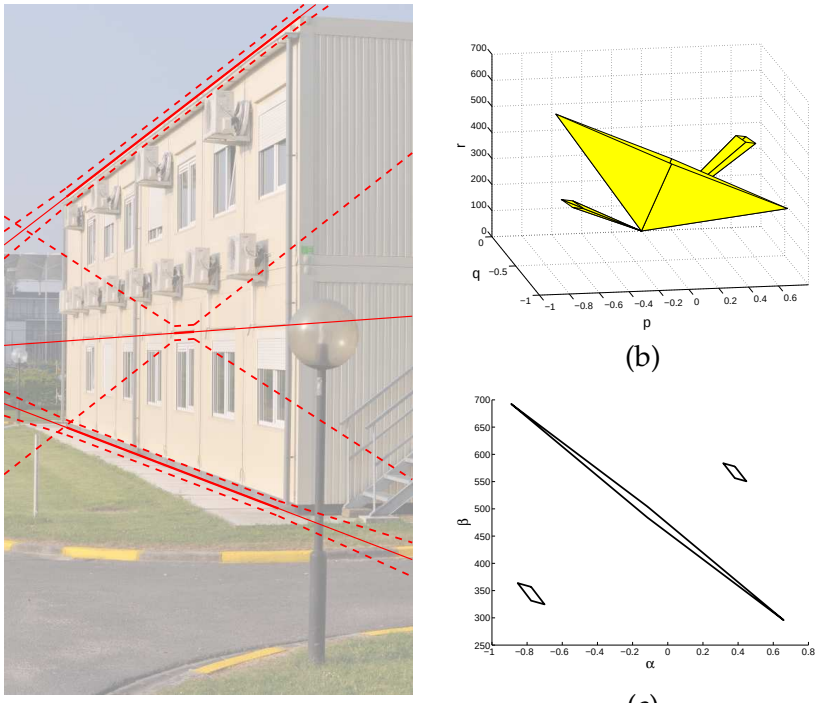
Figure 5.21 shows an example for straight lines in an image. Note that the PUP for longer line segments is smaller than that for shorter line segments.

5.5.2 Line Transformations

When the transformation T is given in the form (5.2) for point correspondences, we can easily compute $S'(S, \mathsf{T})$ and $S(S', \mathsf{T})$ for line parameters $\mathbf{l} = [p, q, r]^T$ by applying either B to $\mathbf{l}$, i.e., $\mathbf{l}' = \mathsf{B}\mathbf{l}$,

$$\begin{bmatrix} p' \\ q' \\ r' \end{bmatrix} = \begin{bmatrix} a_5 & -a_4 & 0 \\ -a_2 & a_1 & 0 \\ a_2a_6 - a_3a_5 & a_3a_4 - a_1a_6 & a_1a_5 - a_2a_4 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \quad (5.11)$$

or its inverse. The set of all nonzero scalar multiples of the transformed line parameters (p', q', r') is denoted as $\mathsf{T} < \mathbf{l} >$.



(a) (b) (c)

Figure 5.21: (a) The PURs for lines in the image domain. (b) A PUP in the 3-dimensional parameter space represents the uncertainty of the line parameters for the 3 indicated lines. (c) If $q \neq 0$, the line can also be represented as $y - \alpha x - \beta = 0$, and then the uncertainty is denoted by polygonal regions in the $\alpha\beta$ parameter space.

Let $\mathbf{l_1}, \mathbf{l_2}$ be the parameter vectors of two lines. For any linear combination of line parameters we have $\mathsf{B}(\alpha_1\mathbf{l_1} + \alpha_2\mathbf{l_2}) = \alpha_1\mathsf{B}\mathbf{l_1} + \alpha_2\mathsf{B}\mathbf{l_2}$. This has an immediate consequence, similar to Proposition 5.2.

Proposition 5.7 *Let* T *be an affine transformation. The transformation of a convex combination of line parameters is equal to the convex combination of the transformed line parameters. That is,* $\mathsf{T} < \alpha_1\mathbf{l_1} + \alpha_2\mathbf{l_2} >= \alpha_1\mathsf{T} < \mathbf{l_1} > +\alpha_2\mathsf{T} < \mathbf{l_2} >$, *with* $\alpha_1 + \alpha_2 = 1$, *and* $0 \leq \alpha_1, \alpha_2 \leq 1$.

According to Proposition 5.7, a PUP C for the line parameters is transformed by T into the PUP C'. To find C' it is sufficient to transform the vertices of a polytope that generates C.

Likewise, given C' one can find C, by applying T^{-1}. Then we can compute $\mathcal{C}'(\mathcal{C}, \mathsf{T})$ as well as $\mathcal{C}(\mathcal{C}', \mathsf{T})$ for a collection of PUPs $C_i \in \mathcal{C}$

and $C'_i \in \mathcal{C}'$. Thus we can solve the RoI problem with known camera parameters almost as easily for lines as for points.

5.5.3 Uncertainty of Line Transformations

When considering the uncertainty on the transformation of straight lines, we can distinguish two different approaches. We can look at the problem as such for lines only, which is relatively straightforward. However, we want to consider the uncertainty on transformations from uncertain point and line position parameters in *one* transformation parameter space $a_1 \dots a_6$, which requires additional care.

The major difficulty in the derivation of line transformation uncertainty is that a TUP for the transformation parameters as in Eq. 5.2 does not correspond to a TUP for the elements b_i of the matrix B (Eq. 5.11). Thus, although B transforms line parameters in a linear way, a convex combination of affine transformations $\alpha_1\mathsf{T}_1 + \alpha_2\mathsf{T}_2$ does not correspond to a convex combination $\alpha_1\mathsf{B}_1 + \alpha_2\mathsf{B}_2$ of transformations which transform line parameters into line parameters. Consequently, for line parameters there is no equivalent of Proposition 5.3.

When combining both point and line features in the same framework for one transformation parameter domain $a_1, \dots, a_6$, the set of transformations can in general *not* be described by a set of linear inequalities. We shall consider a solution for this problem in two different approaches. First, we will look at an important special case with more constrained transformations, where an equivalent for Proposition 5.3 exists and the transformed line parameters do form a polytope. Second, we shall look at the general, and more difficult general case of solving the problem in one parameter domain $a_1, \dots, a_6$, where it is necessary to introduce an approximation for the convex sets of transformations.

Constrained Transformations

In general, the transformed line vector $(p', q', r')^T$ of form (5.11) is not a linear expression in the transformation parameters $a_1, \dots, a_6$ unless some of the coefficients are either vanishing or fixed. To examine how linearity is violated, we construct a conflict graph where the vertices represent the parameters $a_1, \dots, a_6$ of the affine transformation. The conflict graph in Figure 5.22 is constructed by connecting two parameters by an edge when they should not occur together as free parameters in the expressions of (5.11) in order to achieve linearity.

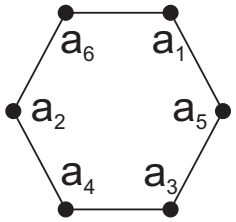


Figure 5.22: Conflict graph for line parameters after affine transformation. When 2 parameters should not occur together as free parameters in the expressions (5.11) in order to achieve linearity, they are connected by an edge.

An independent set is a set of parameters for which each pair is not connected by an edge in the conflict graph, e.g., $\{a_1, a_2, a_3\}$ and $\{a_4, a_5, a_6\}$. Each independent set leads to a special case for which the transformed line parameters (5.11) are expressed as a linear expression with the transformation parameters $a_1, \ldots, a_6$.

The most interesting case consists of transformations of the form

$$\mathsf{A}_{1136} = \begin{bmatrix} a_1 & 0 & a_3 \\ 0 & a_1 & a_6 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.12}$$

which involves uniform scaling and translation. This is not an independent set, but there is a way out. The transformed line vector (p', q', r') is $(a_1 p, a_1 q, -a_1 a_3 p - a_1 a_6 q + a_1^2 r)$. Since any multiple of a line vector represents the same line, we can eliminate a_1, to obtain the image vector $(p, q, -a_3 p - a_6 q + a_1 r)$, which are linear expressions in a_1, a_3, a_6. We therefore have the following proposition.

Proposition 5.8 *Let* $\mathsf{T}_1, \mathsf{T}_2$ *be affine transformations of the form* A_{1136} *(both transformations must be of the same form). Let* $\mathbf{l}$ *be a column vector of line parameters. Then the transformed line parameters of a convex combination of transformations are equal to the convex combination of the transformed line parameters. That is,* $(\alpha_1 \mathsf{T}_1 + \alpha_2 \mathsf{T}_2) < \mathbf{l} >= \alpha_1(\mathsf{T}_1 < \mathbf{l} >) + \alpha_2(\mathsf{T}_2 < \mathbf{l} >)$*, with* $\alpha_1 + \alpha_2 = 1$*, and* $0 \leq \alpha_i \leq 1$*.*

When Proposition 5.8 holds, $\mathcal{T}(S, \mathcal{R}')$ and $\mathcal{R}'(S, \mathcal{T})$ can be solved by similar computations as for points.

Figure 5.23 illustrates the computation of a TUP when the transformations are of the form A_{1136}. We must compute a TUP for a given set of line vectors $\mathbf{l_i}$, and their corresponding PUPs C_i', as given in Figure

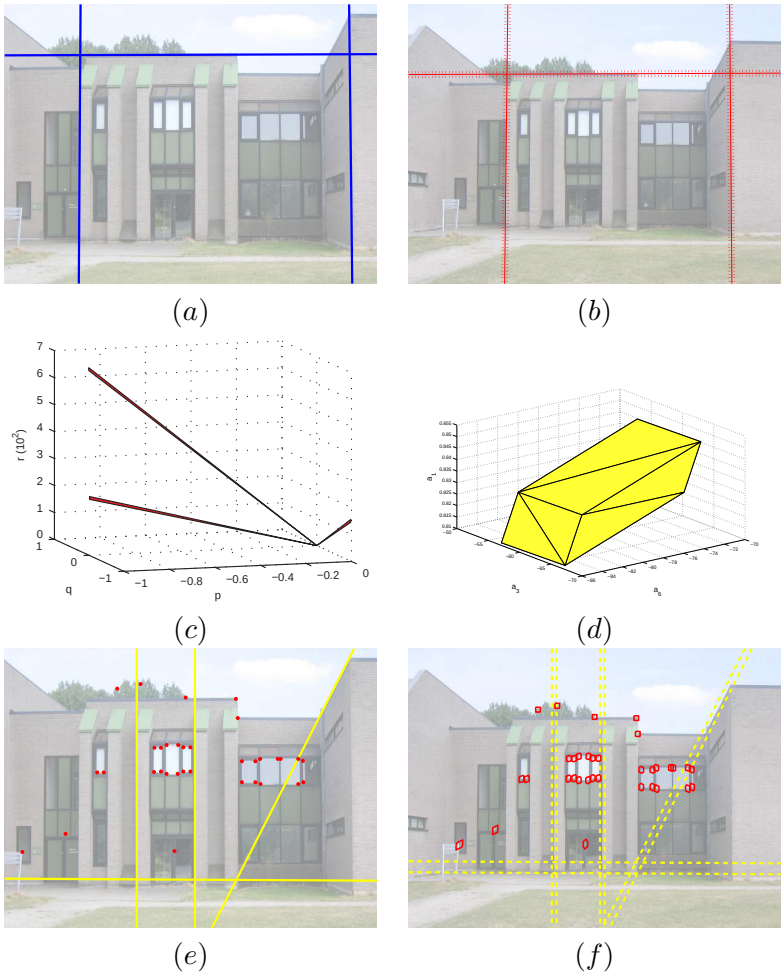


Figure 5.23: A TUP of constrained affine transformations can be computed for the set S of lines $\mathbf{l_i}$ (a) and their corresponding PURs $R'_i \in \mathcal{R}'$ (b). (c) The PUPs C'_i for each of these uncertainty regions in the parameter space pqr. (d) The TUP contains all transformations T_{ij} from $\mathbf{l_i}$ to $\mathbf{l'_{ij}} \in C'_i$ in the parameter space $a_1a_3a_6$. $(e-f)$ The transformations in the TUP map the lines in (e) to their corresponding RoIs in (f). The TUP can also be used to map points (e) to their corresponding RoIs (f).

5.23 $(a-c)$. Note that when the transformations are of the form A_{1136}, then all the lines $\mathbf{l'_{ij}}$ in the PUP must have the same slope as the line $\mathbf{l_i}$, as illustrated in Figure 5.23 (c).

For each C'_i, let $\mathbf{l'_{ij}}$ be a finite set of line parameters such that C'_i is

the smallest pyramid that contains the line parameters $\mathbf{l'_{ij}}$ and the origin, i.e., the $\mathbf{l'_{ij}}$ are the edges of the polyhedron generating C'_i. Then compute a transformation T_{ij} of the form A_{1136} for each subset of correspondence pairs $(\mathbf{l_i}, \mathbf{l'_{ij}})$. The convex hull of all T_{ij} gives the TUP in the transformation parameter space $a_1a_3a_6$, which is illustrated in Figure 5.23 (d).

To compute RoIs in the second image for other line features $\mathbf{l_o}$, we compute $\mathsf{T}_i < \mathbf{l_o} >$ for each vertex T_i of the TUP. The PUP for the transformed image of a line $\mathbf{l_o}$ is the smallest PUP that contains all the points $\mathsf{T}_i < \mathbf{l_o} >$ and the origin. Figure 5.23 (f) shows the uncertainty regions for the lines indicated in (e), computed with the TUP (d).

One of the advantages of deriving the TUP in this manner is that these parameters are also applicable in point feature transformations. As indicated in Figure 5.23 $(e - f)$, the mapping of points according to the same TUP results in bounded polygons as RoIs.

Unconstrained Transformations

This section considers the general case for transformations of straight lines. What is the result of applying a convex combination of affine transformations to a vector of line parameters, when there are no special conditions imposed on the transformations?

Proposition 5.9 *Let* $\mathsf{T}_1, \mathsf{T}_2$ *be two affine transformations, and let* $\mathsf{T} = t\mathsf{T}_1 + (1-t)\mathsf{T}_2$, *with* $0 \leq t \leq 1$, *denote the transformations in their convex span. Let* α, β *be the parameters of the line* $y = \alpha x + \beta$. *Then the parameters* α', β' *of the transformed line lie on a common conic, for* $0 \leq t \leq 1$.

Figure 5.24 (a) shows the result of transforming the line $y = x/3 - 5/4$, with parameter vector $\mathbf{l} = (1/3, -1, -5/4)$ by a convex set of transformations T. The transformations T belong to the convex span of transformations $\mathsf{T} = \sum_i \gamma_i \mathsf{T}_i$ with coefficients $(a_1, a_2, a_3, a_4, a_5, a_6) = \gamma_1(2,2,4,4,3,1) + \gamma_2(2,3,4,4,3,4) + \gamma_3(1,3,3,2,4,4) + \gamma_4(3,2,2,4,2,4) + \gamma_5(1,3,4,2,4,1)$, with $\sum_i \gamma_i = 1$ and $0 \leq \gamma_i \leq 1$.

Figure 5.24 (a) shows the range of the parameters (α', β') of the transformed line $y = \alpha' x + \beta'$. The conic sections correspond to parameters transformed by convex spans of the form $\gamma_i \mathsf{T}_i + \gamma_j \mathsf{T}_j$, $1 \leq i < j \leq 5$. Thus they correspond to transformations that lie along an edge of the boundary of the convex set of transformations. The lines transformed

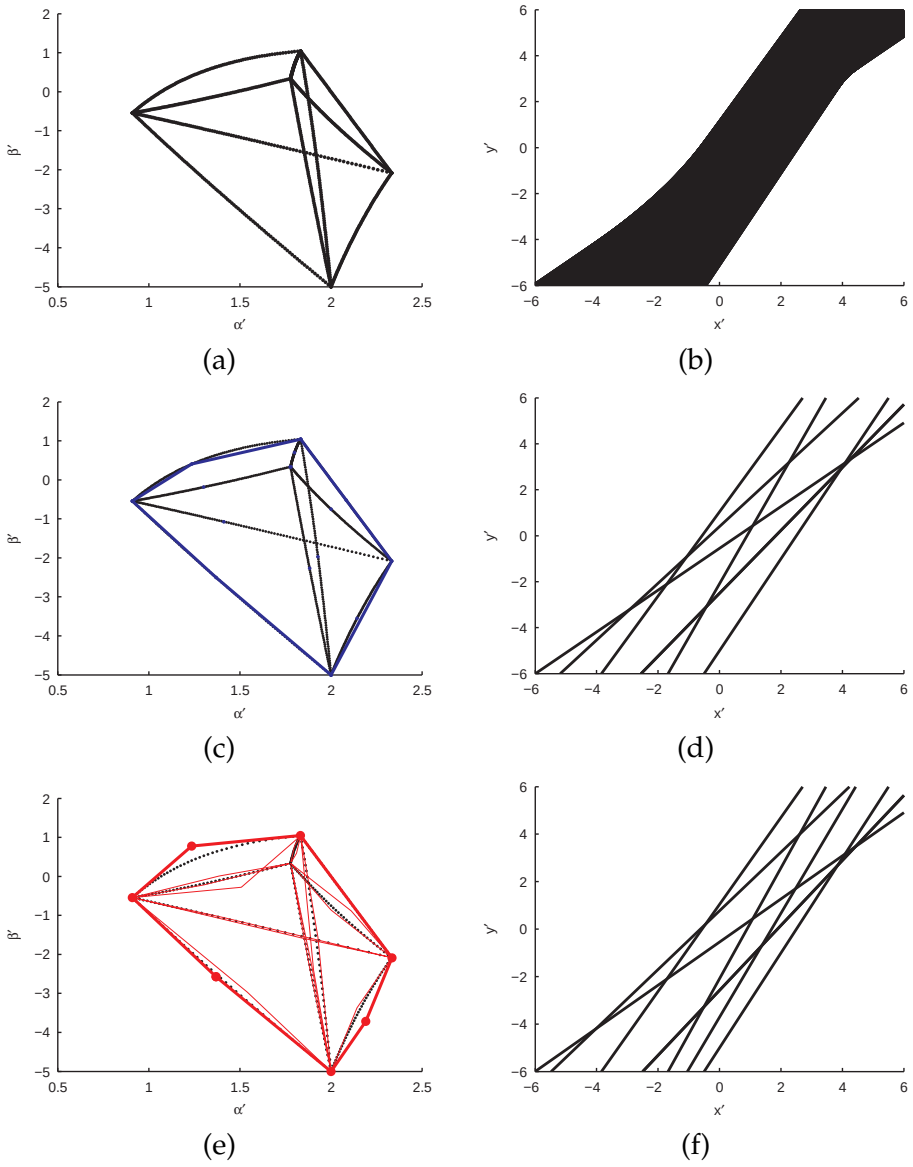


Figure 5.24: An approximation for the transformed line parameter set in (a) is obtained by two approaches. (c) Take the convex hull of a selection of parameter points on the conic segments. (e) Take the convex hull of all enclosing quadrangles for each of the conic segments. (b) The transformed lines in the image domain for each line parameter indicated in (a). The transformed lines corresponding to the vertices of the convex hull in (c, e) are shown resp. in (d, f).

according to those transformations are shown in the image domain in Figure 5.24 (b).

Unconstrained Transformations in Practice

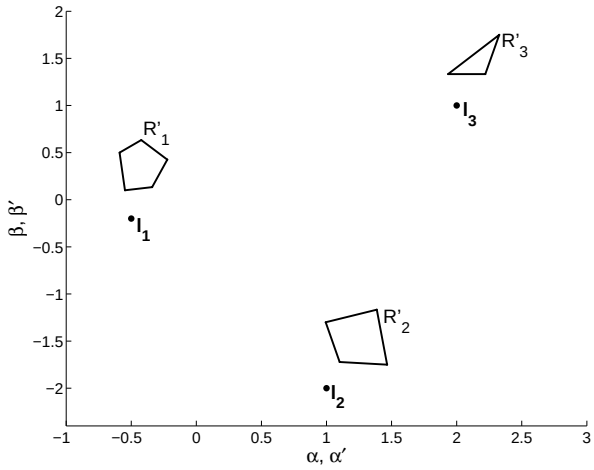
Although Proposition 5.9 shows that the RoI problem for lines is considerably more difficult in the general case, it gives some hints about how we can proceed in practice. In real applications, there are several ways to approach the computation of a convex polytope that approximates the set of transformed line parameters. We can also give an approximation for the set of transformations as a TUP.

Approximation for the set of transformed line parameters. To solve the problem $R'(\mathbf{l}, \mathcal{T})$, one can obtain a convex set for the transformed line parameters. We will shortly present 2 approaches, for which the details are given in [Teelen and Veelaert, 2009].

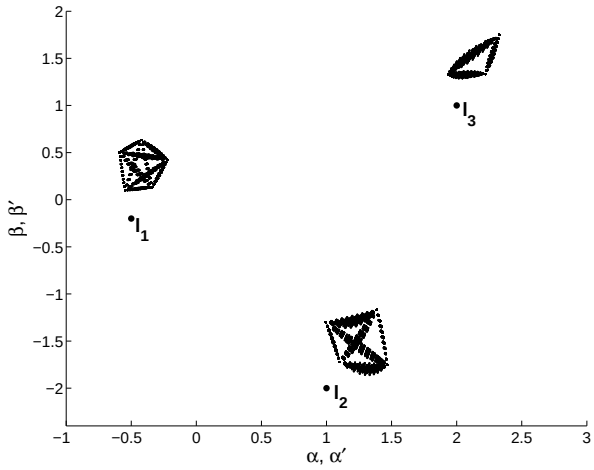
First, one can obtain a close approximation of the uncertainty set of the transformed lines by a polygonal approximation for the boundaries of the set of transformed line parameters. Figure 5.24 (c) shows the simple result of taking the convex hull of the transformed line parameters $\mathsf{T}_i < \mathbf{l} >$, as well as $((\mathsf{T}_i + \mathsf{T}_j)/2) < \mathbf{l} >$. The approximation is shown in solid lines over the dashed lines, which represent the set of line parameters. This approach produces a set that is not necessarily an enclosing set for the transformed line parameters.

Second, if we want to obtain an *enclosing* convex set for the transformed line parameters, it suffices to compute an enclosing quadrangle for each conic segment, and to take the convex hull of all the quadrangles. The approximating convex hull (in solid lines) encloses all transformed line parameters (in dashed lines) in Figure 5.24 (e). The set of transformed lines in the image domain are in both cases (Figure 5.24 (d, f)) a good approximation of the situation in Figure 5.24 (b).

Approximation for transformation sets. We now consider the computation of a transformation uncertainty set for the mapping of a set of lines onto their corresponding uncertainty regions of line parameters. Suppose we have three lines $\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}$ and three uncertainty polygons R'_1, R'_2, R'_3 in the line parameter space, we are looking for the transformations T that map each line $\mathbf{l_i}$ onto a line whose parameters lie in the polygon R'_i. This situation is illustrated in Figure 5.25 (a).



(a)



(b)

Figure 5.25: (a) The parameters (α_i, β_i) of 3 lines $\mathbf{l_i}$ $(y - \alpha_i x - \beta_i = 0)$ and 3 PURs R'_i. We compute a set of transformations T_{ijk} from the correspondences of each $\mathbf{l_i}$ and all subsets of 3 vertices on the corresponding R'_i. $\tilde{W}$ is the convex hull of all T_{ijk}. (b) The parameter sets when the lines are transformed by the polytope $\tilde{W}$.

According to Proposition 5.9, convex combinations of transformations yield conic sections for the transformed line parameters. Similarly, when mapping the line parameters of a line $\mathbf{l_i}$ on the set of parameters from a convex combination of line parameters, the resulting set of transformations W is not a convex combination of transformations. To illustrate, let $\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}$ be the three lines, and let $\mathbf{v_{1i}}, \mathbf{v_{2j}}, \mathbf{v_{3k}}$ be the lines that correspond to the vertices of the line uncertainty polygons R'_1, R'_2, R'_3. Let T_{ijk} be the affine transformation mapping $\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}$ onto $\mathbf{v_{1i}}, \mathbf{v_{2j}}, \mathbf{v_{3k}}$, for one particular choice of three vertices, one from each polygon. For example, if each uncertainty region is a triangle, there will be 27 transformations T_{ijk}.

As the set of all transformation T_{ijk} is not a polytope, we resort to an approximation for the set W. Proposition 5.9 gives more insight into the properties of this approximation. Let $\tilde{W}$ be the convex hull of the transformations T_{ijk}. In general, the polytope $\tilde{W}$ will not be the same as the set W. However, if we compute for each line $\mathbf{l_i}$ the line parameters as transformed by the transformations in $\tilde{W}$, we can compare these parameter regions $\tilde{R}'_i$ with the given regions R'_i. If the transformed regions $\tilde{R}'_i$ are close to R'_i, then $\tilde{W}$ will be a good approximation for W.

Figure 5.25 (a) shows the parameters of three lines and three uncertainty regions. Figure 5.25 (b) shows the parameter sets when the lines are transformed by the polytope $\tilde{W}$. Since the transformed parameters lie close to the given regions R'_i, the transformation set $\tilde{W}$ is a good approximation for W.

5.5.4 Computing Regions of Interest

We will illustrate the solution for the RoI problem for affine transformations of a combination of points and lines in an example involving the tracking of traffic signs in a video sequence from a dynamic camera mounted in a vehicle. We know that most traffic signs can be described by a few remarkable features, lines or points. We assume that the transformation of the set of features from one frame to the next can be described by an affine transformation. The signs are perpendicular to the driving direction, and the time span in between 2 frames is short, so that perspective effects play a small role.

If a sufficient number of corresponding lines is detected in consecutive frames, solve the RoI problem by computing the transformation for that sample set, otherwise add point correspondences to the sample

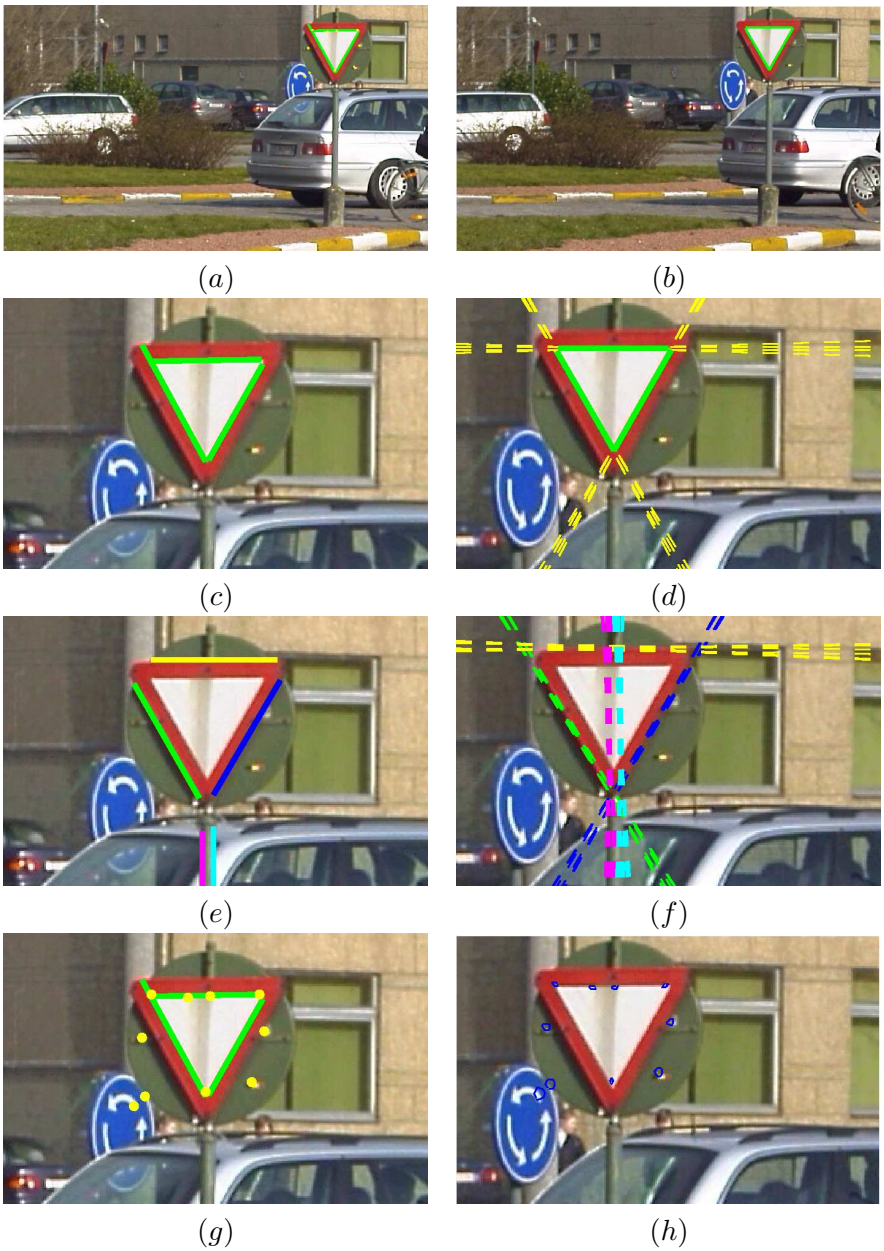


Figure 5.26: Computing RoIs for affine transformation models in tracking applications. (a) and (b) respectively show frame(t) and frame(t+1) of the sequence. $(c-d)$ An sample set of 3 line correspondence pairs. A TUP is computed from these correspondences and used to compute the IURs for line and point features in (e) and (g). The resulting RoIs are indicated in (f) and (h).

set. Figures 5.26 $(c-d)$ show the corresponding lines detected by the Radon transform in frame t_i and t_{i+1}. From the line correspondences and the estimated uncertainty (d), we compute the TUP.

The computed TUP can then be used to obtain accurate RoIs for a set of other line features, not in the sample set. Some illustrative examples of these RoIs are shown in Figure 5.26 $(e-f)$. The same TUP can then be used to solve the RoI problem for point correspondences, as shown in Figure 5.26$(g-h)$.

5.6 Conclusion

We presented an uncertainty framework that models location uncertainty by polygonal regions in the parameter space, both for line and point features. The positional uncertainty propagates naturally to the uncertainty on the parameters of a transformation. We will show in Chapter 7 that the uncertainty framework can be extended toward the more general 2D projective transformations.

Our uncertainty framework fits nicely into a RANSAC approach, where one must solve the RoI problem in each step of a matching procedure. The main advantage is that the uncertainty on possibly corrupted location parameters of features is modeled so that not only one transformation with propagated error is considered, but a polytope of transformation parameter points. This will be the subject of the next chapter.

Original contributions. The development of the transformation uncertainty framework is described in [Teelen and Veelaert, 2005b], [Teelen and Veelaert, 2004a], and [Teelen and Veelaert, 2004b]. The use of positional uncertainty in the computation of RoIs for geometric features is discussed in [Teelen and Veelaert, 2009] and [Teelen and Veelaert, 2008].

Chapter 6

Consensus

The previous chapter concentrates on an important step in the solution of the matching problem: the computation and use of the implied regions as RoIs in detecting candidate matches for features. This chapter shows how to fit that step into a robust estimation procedure to determine a consensus set of correspondences. Once the spatially consistent correspondences are established, the next step is to find a transformation that optimally maps a first set of features onto their correspondences.

6.1 Introduction

A general 2D correspondence problem can be described as: *Given* a finite set S of image points $\mathbf{x} \in \mathbb{R}^2$, and a finite set S' of candidate matching points $\mathbf{x}' \in \mathbb{R}^2$, *find* the *optimal* transformation $\mathsf{T}_{\mathsf{opt}}$ that maps points $\mathbf{x_i} \in S$ on corresponding image points $\mathbf{x'_i} \in S'$.

The solution for such a matching problem consists of two steps. First, a robust estimation procedure must determine a correspondence map relating the $\mathbf{x_i}$ to their correct matches $\mathbf{x'_i}$. In each iteration, an estimate for the map is derived from a small sample set of randomly chosen correspondences. Each estimated map has an associated *consensus set* of inliers, i.e., the set of correspondences that comply with the estimated map. The procedure finally returns the best consensus set according to some criterion, which should preferably coincide with the true correspondence set. This set must be discriminated from other, false candidates, i.e., the outliers. As the localization uncertainty of the

features could interfere with this process, we will show in the next section how our uncertainty procedure fits into a robust estimation framework.

The second step involves the estimation of the transformation that optimally relates the correspondence pairs in the resulting consensus set. Note that the solution for a correspondence problem might not be unique. For instance, in the case of repetitive patterns (e.g. a checkerboard) there are several equally acceptable candidate matches for each feature in the first image. Then there might be more than one solution, as multiple transformations could minimize some cost criterion. This case could occur in practice when using an iterative procedure to solve the correspondence problem, certainly when starting from a small set of features.

In section 6.3, we will present a criterion to obtain an optimal transformation by the minimization of uncertainty in the spatial domain.

6.2 Establishing the Correspondence Map

To enhance a robust estimation procedure with our uncertainty framework, we insert our uncertainty model in the backbone of a typical RANSAC process, and denote it as U-RANSAC. The main difference is the estimation of the transformation from a sample correspondence set by a TUP, rather than using only one estimate as in the standard approach.

6.2.1 Consensus set

We first show how to determine which correspondence pairs are in the consensus set $C(\mathcal{T})$ of a TUP $\mathcal{T}$. $C(\mathcal{T})$ is the set of correspondence pairs for which candidate matches $\mathbf{x}'_\mathbf{j}$ in the second image are located in the IUR $R'(\mathbf{x_i}, \mathcal{T})$ of a feature $\mathbf{x_i}$.

Definition 6.1 *Let S be a finite set of points $\mathbf{x_i}$, and S' be a finite set of candidate matches $\mathbf{x}'_\mathbf{j}$. Then the consensus set $C(\mathcal{T})$ of the TUP $\mathcal{T}$ contains all pairs $\{(\mathbf{x_i}, \mathbf{x}'_\mathbf{j})\}$ for which $\mathbf{x}'_\mathbf{j} \in R'(\mathbf{x_i}, \mathcal{T})$.*

Note that a consensus set may list some points of S more than once, e.g., we may have $C(\mathcal{T}) = \{(\mathbf{x_1}, \mathbf{x}'_\mathbf{1}), (\mathbf{x_1}, \mathbf{x}'_\mathbf{2}), \ldots, (\mathbf{x_2}, \mathbf{x}'_\mathbf{2}), \ldots\}$. For

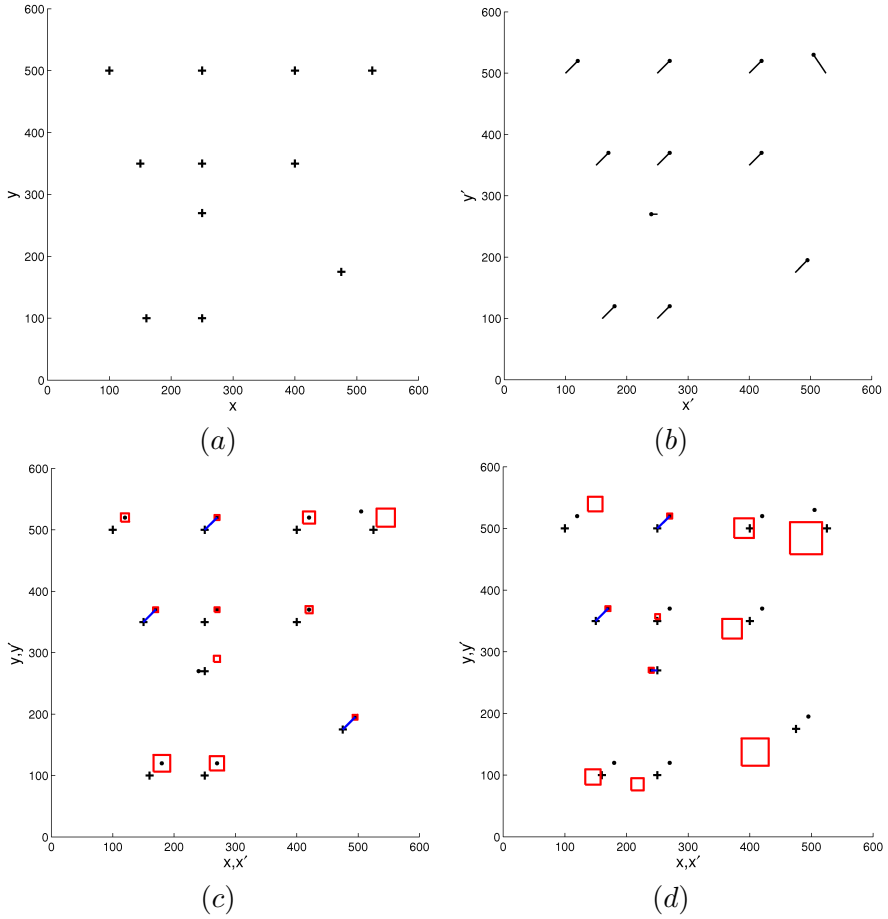


Figure 6.1: $(a-b)$ The features $\mathbf{x_i} \in S$ (crosses in (a)) and their putative correspondences $\mathbf{x'_i} \in S'$ (dots in (b) with an indication of the map). $(c-d)$ Compute $\mathcal{T}_1$ (c) and $\mathcal{T}_2$ (d) for the indicated correspondence pairs (where both S and S' are given in the same axes). After computing the IURs $R'(\mathbf{x_i}, \mathcal{T}_1)$ and $R'(\mathbf{x_i}, \mathcal{T}_2)$ (rectangles in $(c-d)$) for all points $\mathbf{x_i} \in S$, we obtain consensus sets with respective consensus measures $n(C(\mathcal{T}_1)) = 9$ (c) and $n(C(\mathcal{T}_2)) = 3$ (d).

a consensus set $C(\mathcal{T})$ we define a *consensus measure* $n(C(\mathcal{T}))$, which counts the number of points in S that occur at least once in $C(\mathcal{T})$.

Definition 6.2 *Let A be the largest subset $\{\mathbf{x_i}, \ldots\} \subseteq S$, such that for each point $\mathbf{x_i}$ in A, there is at least one pair of the form $(\mathbf{x_i}, \mathbf{x'_j})$ in C. The consensus measure $n(C(\mathcal{T}))$ is the cardinality of the set A.*

When given a set S of points $\mathbf{x_i}$ and a set S' of candidate matches $\mathbf{x'_i}$, we first choose a sample subset $P_s = \{(\mathbf{x_i}, R'_i), (\mathbf{x_k}, R'_k), \ldots\}$, with a PUR $R'_i \in \mathcal{R}'_s$ for each point $\mathbf{x'_i} \in S'_s$. From the sample subset, we compute a TUP $\mathcal{T} = \mathcal{T}(S_s, \mathcal{R}'_s)$. The sample subset must then be part of the consensus set, i.e., $P_s \subseteq C(\mathcal{T}(S_s, \mathcal{R}'_s))$, provided $\mathcal{T}(S_s, \mathcal{R}'_s) \neq \emptyset$. The consensus measure is augmented by one for each IUR containing at least one candidate match $\mathbf{x'_i}$.

We can use the consensus measure as a parameter to judge the reliability of a transformation hypothesis in a RANSAC-like procedure. The higher the measure, the more reliable the model is (hereby avoiding degenerate configurations of correspondences).

Figure 6.1 shows an example for two different TUPs $\mathcal{T}_1$ (b) and $\mathcal{T}_2$ (c), computed from the correspondence pairs connected by a solid line. Notice that $n(C(\mathcal{T}_1)) > n(C(\mathcal{T}_2))$, thus we can conclude that $\mathcal{T}_1$ is the more reliable of both models, that is, $\mathcal{T}_1$ is the best fit to the data based on the consensus measure.

6.2.2 U-RANSAC

Our approach explicitly incorporates the localization uncertainty of features in a robust estimation procedure, which gives some advantages over the standard framework. Therefore, U-RANSAC is better suited than the common approach in the use of information from good, but inaccurately localized samples. This will prove to be beneficial for our matching procedure (certainly when using small feature sets), as none of the information of less accurate samples is neglected.

We will now sketch the general outline of the U-RANSAC algorithm, which iteratively estimates a map from S to $\mathcal{R}'$. A more detailed discussion can be found in [Veelaert and Teelen, 2006a].

Step 0: Feature extraction. Extract the feature sets S and S'. $\mathcal{R}'$ denotes the set of square PURs R'_i centered on each point in the set S'. The height ε of each R'_i must be chosen in advance, by estimating the uncertainty in the application (e.g. the maximal error made by the feature detector).

Step 1: Selection of sample correspondence pairs. Randomly select a sample subset P_s of correspondences $(\mathbf{x_j}, R'_j)$ of sufficient size (e.g. 3 pairs for the affine transformation) and in general position. Generate a hypothesis about the map by computing the trans-

formation model as the TUP $\mathcal{T}_s(S_s, \mathcal{R}'_s)$ from the correspondence pairs in P_s. Take the remarks in section 3.3.1 into account when selecting P_s.

Step 2: Determine the consensus set. Compute the IURs $\mathcal{R}'(S, \mathcal{T}_s)$ for all points in S. Determine the *consensus set* $C(\mathcal{T}_s)$ by adding all putative correspondence pairs $(\mathbf{x_j}, \mathbf{x'_k})$ for which the IUR $R'(\mathbf{x_j}, \mathcal{T}_s)$ intersects with the PUR R'_k.

Step 3: Stopping criterion. Verify whether the consensus set obtained for the current hypothesis meets the requirements of an a priori defined stop criterion. If so, proceed to Step 4. Else, go back to Step 1. For example, terminate if a sufficient number of correspondences is returned, i.e., verify whether the consensus measure $n(C(\mathcal{T}_s))$, or the ratio $n(C(\mathcal{T}_s))/|S'|$, exceeds some threshold (e.g. the expected number of inliers). Or alternatively, stop when the number of performed iterations is larger than some threshold.

Step 4: Determine the optimal transformation. If the stop criterion is met, extract an optimal transformation from the resulting consensus set.

In the common RANSAC approach, the actual location of the features $\mathbf{x'_i}$ is compared to the transformed features $\mathsf{T_s}(\mathbf{x_i})$ where $\mathsf{T_s}$ is estimated from the sample set (S_s, S'_s). A fixed distance threshold is applied to discriminate a consensus set $C(\mathsf{T_s})$. The localization of features can be slightly disturbed by errors during feature detection. These localization errors propagate into the estimate of the transformation during the calculations of the hypothesis step. We explicitly take the localization uncertainty into account, and compute a set of transformations $\mathcal{T}_s(S_s, \mathcal{R}_s)$ instead of one unique transformation $\mathsf{T_s}(S_s, S'_s)$. Thus, in Step 2 we must consider the propagation of uncertainty during the computation of the consensus set (i.e. in the mapping of all features in S).

In the comparison of Figure 6.2, we mimic that behavior of the common RANSAC verification strategy by choosing a square region of fixed height [Veelaert and Teelen, 2006a]. The height of these regions is chosen to match the expected uncertainty introduced by the feature detector. Although our framework can cope with arbitrary shaped PURs, rectangular regions (with sides aligned with the image axes) provide computational benefits for the practical implementation.

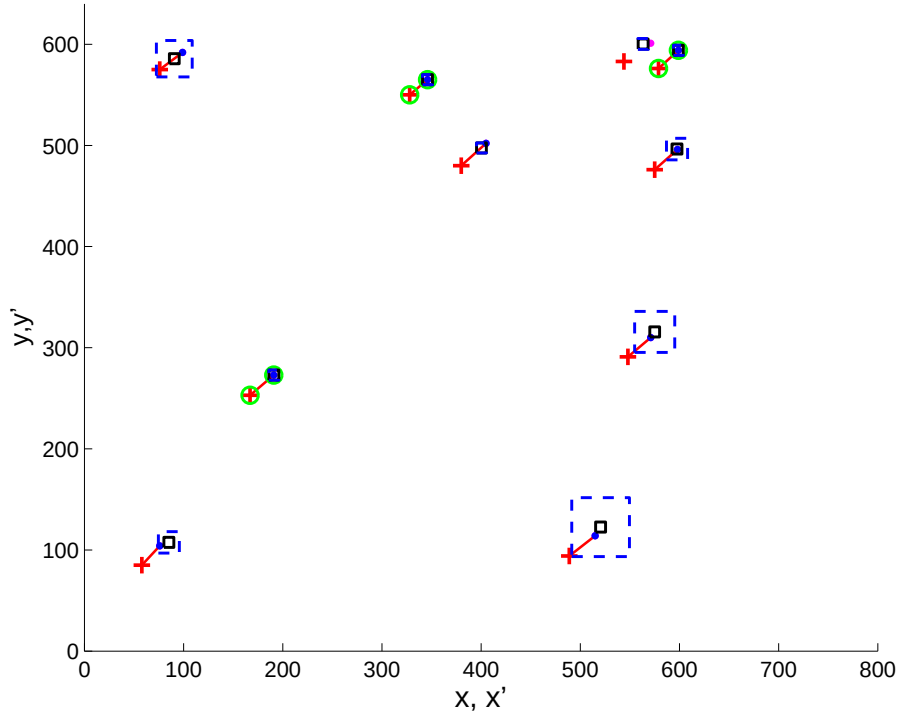


Figure 6.2: A comparison of U-RANSAC to RANSAC. Find the transformation from the features $\mathbf{x_i} \in S$ (+) to the features $\mathbf{x'_i} \in S'$ (•), as indicated by the solid line. The hypothesis for the transformation is computed from the sample set of the 3 pairs indicated by circles. In U-RANSAC, the IURs (dashed lines) of varying size determine the consensus set, while for the common RANSAC approach, the regions (solid lines) are of a fixed height.

In our framework, the positional uncertainty is propagated into the computation of the IURs, which are in general larger than the regions in the common approach (see the dashed rectangles in Figure 6.2). Then, the consensus set $C(\mathcal{T}_s)$ computed for a corrupted sample set P_s contains on average more (and correct) correspondences. This allows the U-RANSAC process to terminate after fewer iterations than a standard approach. There is of course an additional cost of propagating the uncertainty, albeit a limited effort because efficient methods for computing the variation in size of rectangular IURs (see section 5.4.2), and the intersection of rectangles are available.

To support this intuition we conducted a Monte Carlo simulation.

In each experiment a ground truth set of n correspondences is established for a randomly chosen transformation T relating the $\mathbf{x_i}$ to $\mathbf{x'_i}$ (i.e. the ground truth set). We added a given percentage of outliers, (i.e. features for which $\mathbf{x_j} \neq \mathsf{T}(\mathbf{x'_j})$), and superposed localization errors on both the $\mathbf{x_i}, \mathbf{x_j}$ and the $\mathbf{x'_i}, \mathbf{x'_j}$.

These experiments show that U-RANSAC discriminates the ground truth correspondence set on average after fewer iterations (up to 7 times less) than the common approach. This intuition is confirmed by the results of the Cov-RANSAC algorithm [Raguram et al., 2009], where a different model for uncertainty is incorporated in the RANSAC framework, when the probability of an all-inlier sample set seems adequate. Cov-RANSAC achieves up to a 10-fold reduction in the number of samples evaluated in geometric estimation problems. From both Cov-RANSAC and U-RANSAC, we can conclude that it is beneficial to include uncertainty information in the robust model estimation process, both in hypothesis and the verification stage.

6.2.3 Conclusion

When the features are expected to be inaccurately localized, we advise to incorporate the localization uncertainty in a RANSAC-like approach for a matching procedure. However, some problems still remain. It is required to set parameters by prior estimations, e.g., for the expected number of inliers, which we want to avoid. For small feature sets S and S', we want to give a reliability measure for an obtained consensus set, other than just counting its size. These considerations will be the subject of Chapter 8.

The definition of the consensus set is not symmetric in S and S'. The symmetrization of the matching problem would involve to also consider the TUP (S', S) in some way, but requires careful attention (considering the results of section 5.4.3). In our opinion, this approach could lead to better results, but its elaboration remains subject for further research.

Other approaches to improve the common RANSAC scheme developed methods to predict how reliable the hypothesis from a sample subset actually is, before its verification. This would also be beneficial in our U-RANSAC approach, because a well-chosen sample subset results in a more restricted TUP, and thus a more reduced search space in the following steps. This will also remain subject for future research.

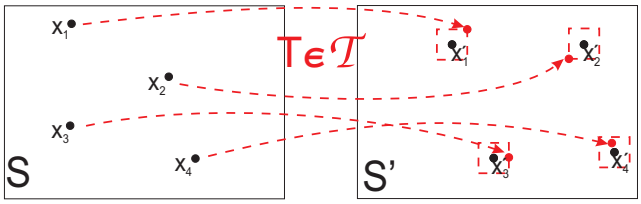


Figure 6.3: Our geometric cost criterion in the determination of an optimal transformation.

6.3 Computing the Optimal Transformation

Once a consensus set $C(\mathcal{T}_s)$ of correspondence pairs $(\mathbf{x_i}, \mathbf{x'_i})$ is returned by the robust estimation procedure, one wants to find a transformation $\mathsf{T}_{\mathsf{opt}}$ that maps each point $\mathbf{x_i} \in S_C$ to a point as close as possible to its matching point $\mathbf{x'_i} \in S'_C$. Hartley and Zisserman have given an overview of geometric cost functions (either in the image plane, in the source plane, or in both planes), that can be used to compute the optimal transformation [Hartley and Zisserman, 2003] (see section 4.5). Instead of computing the generalized inverse or to minimize the average squared distances, we choose the maximal distance $d_i(\mathsf{T}_{\mathsf{opt}}(\mathbf{x_i}), \mathbf{x'_i})$ as our criterion in the minimization problem. This measure returns an adequate indication of the localization uncertainty for the data after transformation in our uncertainty framework.

The optimal transformation $\mathsf{T}_{\mathsf{opt}}$ is derived as one particular transformation in the TUP $\mathcal{T}(S_C, \mathcal{R}'_C)$. $\mathcal{R}'_C$ is a set of rectangular regions centered on the points $\mathbf{x'_i} \in S'_C$, and $\mathcal{T}(S_C, \mathcal{R}'_C)$ describes the set of transformations from all $\mathbf{x_i} \in S_C$ to their corresponding region in $\mathcal{R}'_C$. We demand that the spatial localization uncertainty is minimized, i.e., the regions in $\mathcal{R}'_C$ should be as small as possible, but there must still be (at least) one transformation in the TUP $\mathcal{T}(S_C, \mathcal{R}'_C)$. $\mathsf{T}_{\mathsf{opt}}$ then is that one transformation in the TUP, as illustrated in Figure 6.3. We use σ and ρ to denote respectively the width and height of all rectangular regions centered on the $\mathbf{x'_i}$ in $\mathcal{R}'_C$

For example, for an affine transformation we can describe the problem as two systems P and Q of inequalities for a set S_C of t points

$$P : \begin{cases} |x'_1 - (a_1 x_1 + a_2 y_1 + a_3)| & \leq & \frac{\sigma}{2} \\ \dots & & \\ |x'_t - (a_1 x_t + a_2 y_t + a_3)| & \leq & \frac{\sigma}{2}, \end{cases} \tag{6.1}$$

and

$$Q : \begin{cases} |y'_1 - (a_4 x_1 + a_5 y_1 + a_6)| & \leq & \frac{\rho}{2} \\ \dots & & \\ |y'_t - (a_4 x_t + a_5 y_t + a_6)| & \leq & \frac{\rho}{2}. \end{cases} \tag{6.2}$$

Each inequality expresses that the distance between the coordinates of a transformed point $(a_1 x_i + a_2 y_i + a_3, a_4 x_i + a_5 y_i + a_6)$ and its corresponding image point (x'_i, y'_i) should be smaller than half the length of the boundary of the uncertainty region in the x- and the y-direction, σ and ρ, with $\sigma, \rho \geq 0$. We then must find the transformation parameters for which σ and ρ are minimal.

The systems P and Q can be interpreted in two different ways. First, we determine the transformation parameters $a_1, \dots, a_6$ from the parameters of the parallel planes that fit closest to the point clouds in the (x, y, x') and (x, y, y') parameter spaces. Second, if only the minimal width σ and height ρ of the regions are demanded, we can derive σ and ρ in a more direct way by using an algebraic condition that indicates when there still is a possible non-empty solution for this set. Both approaches will be discussed in more detail below.

6.3.1 Determining the Transformation Parameters

We consider the transformation parameters $a_1, \dots, a_6$ in the systems P and Q (Eq. 6.1-6.2) as the parameters of planes in both the (x, y, x') and (x, y, y') space. The parameters for the optimal transformation $\mathsf{T}_{\mathsf{opt}}$ can then be derived from the parameters of the plane that best fits the point cloud in both 3D spaces. This concept is illustrated in Figure 6.4.

To find $\mathsf{T}_{\mathsf{opt}}$, we look at all pairs of supporting parallel planes for the point clouds. By translating and rotating planes, we can find all planes touching, but not dividing this set of points. There must be at least 4 points of the cloud lying in one of the parallel planes, either 3 on one plane and 1 on the other, or 2 on each plane. Those points are situated so that the planes cannot roll over the cloud of points, i.e., the planes must be lying stable against the cloud when pressure is applied perpendicular to the plane. For the points lying on the parallel planes the equality sign applies, while a strict inequality applies for all other points situated between the parallel planes.

Then the *best* fitting planes are the two planes for which the mutual perpendicular distance is minimal. The geometric interpretation

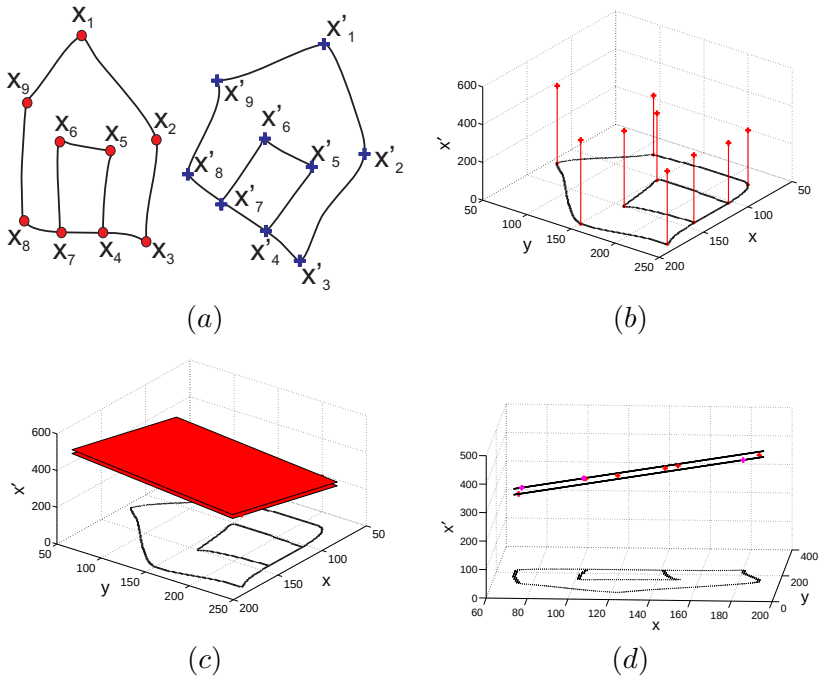


Figure 6.4: The determination of the optimal transformation in a TUP can be related to the fitting of supporting planes to point clouds in a 3D space. (a) We start from a set of corresponding points $(\mathbf{x_i}, \mathbf{x'_i})$. (b) The solution for the system P (Eq. 6.1) is obtained in the space (x, y, x') for the correspondence set of (a). (c) The parallel planes that best fit the point cloud. (d) A view parallel to the best fitting planes.

is simple: these parallel planes enclose the point sets as tight as possible. The approximation for the optimal transformation then consists of the parameters of the plane with equal perpendicular distance to both parallel planes, i.e., the plane lying in the middle.

The influence of each point on the transformation parameters can be investigated by the relative position to one of the two parallel planes. The understanding about how the best fit (and the transformation) is influenced by changes of point positions is relatively simple. A slight displacement of points between the planes will not affect the position of the best fit. If on the other hand 1 of the 4 (or more) points on the best fitting planes is displaced, then this will have a predictable effect on the best fit. Suppose that the point sets still contain an outlier, it can be detected as the point that forces the planes too far apart, and whose

removal results in a considerable reduction in size of the uncertainty regions. If a correspondence set contains several outliers, one could remove them one by one, until the spatial uncertainty is sufficiently reduced, or until the removal of the correspondence does not yield a considerable decrease in region size any more.

6.3.2 Determining the Minimal Region Size

We derive a measure for the minimal width σ and height ρ of the uncertainty regions from a simple algebraic condition which determines whether a system has a solution. One can show that a non-empty solution for the entire system P (or Q) exists if for each subsystem containing 4 inequalities a solution exists [Veelaert, 2003a, 1993]. We will proceed with the subsystems of 4 inequalities from the system P (for Q the derivation is similar).

First, for each subsystem of the system P,

$$\begin{cases} |x'_i - (a_1 x_i + a_2 y_i + a_3)| & \leq & \frac{\sigma}{2} \\ |x'_j - (a_1 x_j + a_2 y_j + a_3)| & \leq & \frac{\sigma}{2} \\ |x'_k - (a_1 x_k + a_2 y_k + a_3)| & \leq & \frac{\sigma}{2} \\ |x'_l - (a_1 x_l + a_2 y_l + a_3)| & \leq & \frac{\sigma}{2}, \end{cases} \tag{6.3}$$

we construct a square 4×4 augmented matrix,

$$(\mathsf{L}|X') = \left(\begin{array}{ccc|c} x_i & y_i & 1 & x'_i \\ x_j & y_j & 1 & x'_j \\ x_k & y_k & 1 & x'_k \\ x_l & y_l & 1 & x'_l \end{array} \right). \tag{6.4}$$

Each row of this matrix consists of the homogeneous coordinates of a point $\mathbf{x_i}$, and in addition, the corresponding x'_i coordinate of the point $\mathbf{x'_i}$ as the fourth row element. Let M_i denote the cofactor corresponding to the element x'_i in the matrix $(\mathsf{L}|X')$, that is, M_i is the signed determinant of the submatrix resulting from the deletion of row i and the fourth column.

We can state a precise condition for the subsystem (6.3) to have a solution provided that matrix $(\mathsf{L}|X')$ has rank 4. The matrix $(\mathsf{L}|X')$ is of rank 4 if at least one of the 4 cofactors is different from zero. In that case the inequality

$$|M_i x'_i + \ldots + M_l x'_l| < (|M_i| + \ldots + |M_l|)\frac{\sigma}{2} \tag{6.5}$$

must hold (as proven in [Veelaert, 1993]). This inequality can be rewritten as

$$\begin{aligned} \frac{\sigma}{2} &> \frac{|M_i x'_i + \ldots + M_l x'_l|}{(|M_i| + \ldots + |M_l|)} \\ &> \frac{|\det(\mathsf{L}|X')|}{|M_i| + |M_j| + |M_k| + |M_l|} \end{aligned} \tag{6.6}$$

using standard linear algebra. For the system Q we obtain

$$\frac{\rho}{2} > \frac{|\det(\mathsf{L}|Y')|}{|M_i| + |M_j| + |M_k| + |M_l|}. \tag{6.7}$$

To summarize, the system P has a solution if each of its subsystems has a solution, which can be verified by first selecting all possible quadruple subsets out of the set of all points. A solution for system P can be found if and only if for each quadruple subset, the inequality (6.6) is satisfied.

If the matrix $(\mathsf{L}|X')$ is not of full rank, the first 3 columns are linearly dependent. By looking further at the dependences, one of the linearly dependent columns can be discarded. Then all 3×3 submatrices obtained by removing one of the rows are taken into consideration, and a similar procedure as for the 4×4 matrix is applied to all 3×3 submatrices. A solution for the system P can be found if a solution can be obtained for each of the 4 submatrices.

As a result, the minimal dimensions (σ, ρ) of the uncertainty regions can be determined by first computing

$$\sigma_{ijkl} = \frac{2|\det(\mathsf{L}|X')|}{|M_i| + |M_j| + |M_k| + |M_l|} \tag{6.8}$$

for all quadruple subsets of the point set S. The minimal dimensions of the uncertainty regions are obtained by taking the maximum value for σ and ρ over these two sets,

$$\sigma = \max_{i,j,k,l} \sigma_{ijkl}, \quad \rho = \max_{i,j,k,l} \rho_{ijkl} \tag{6.9}$$

The values (σ, ρ) yield the minimal dimensions for the uncertainty regions for which there still is a non-empty solution for the TUP. Note that this measure is not necessarily symmetric.

6.3.3 Examples

The above concepts are illustrated in two exemplary applications.

Image Registration

Suppose that a consensus set $C(\mathcal{T})$ is obtained for the example in Figure 6.5. Then we can compute the optimal transformation $\mathsf{T}_{\mathsf{opt}}$ in our framework by the method described above in section 6.3.1. $\mathsf{T}_{\mathsf{opt}} \in \mathcal{T}$ is the transformation that maps the points in S into rectangular uncertainty regions of minimal size centered on the features in S'. For the sets S_C and S'_C given in Figure 6.5 (a) and (b), we obtain the optimal transformation

$$\mathsf{T}_{\mathsf{opt}} = \begin{bmatrix} 0.8271 & -0.0045 & 79.2268 \\ 0.0020 & 0.8297 & 57.3235 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.10}$$

for a region of width 1.94 and height 1.41 as shown in (c).

We can transform all pixel locations according to that $\mathsf{T}_{\mathsf{opt}}$ to obtain a registration of the first image to the second. Figure 6.5 (d) shows the result of the registration, where the transformed first image in the red layer is overlain on the second image in the green layer. If the registration is correct, the overlain part should be yellow, as is the case here.

Similarity Measure

The minimal size of the uncertainty regions for which the TUP is nonempty can serve as a *similarity measure*. If the size of the regions is smaller, less uncertainty or variation is allowed for the localization of the feature in the second set, thus, the more *similar* the feature sets are (apart from a transformation). The definition of this similarity measure and its use in applications for the recognition of different shapes is presented in [Teelen and Veelaert, 2004b] and [Teelen and Veelaert, 2004a].

We assume that multiple similar, but not necessarily identical objects can be characteristically described by their point set. Our approach involves the comparison of a set of points on an unknown object (e.g. characteristic points on the outline or boundary of image objects) to a set of points on representative objects (e.g. a shape database). Then the point set of each new detected shape is compared to each shape in the database, and the most alike must be identified.

As a simple example, consider the hand drawn sketches in Figure 6.6 (a), which are described well by the indicated feature set. We can compare by the similarity measure how alike the feature sets for each

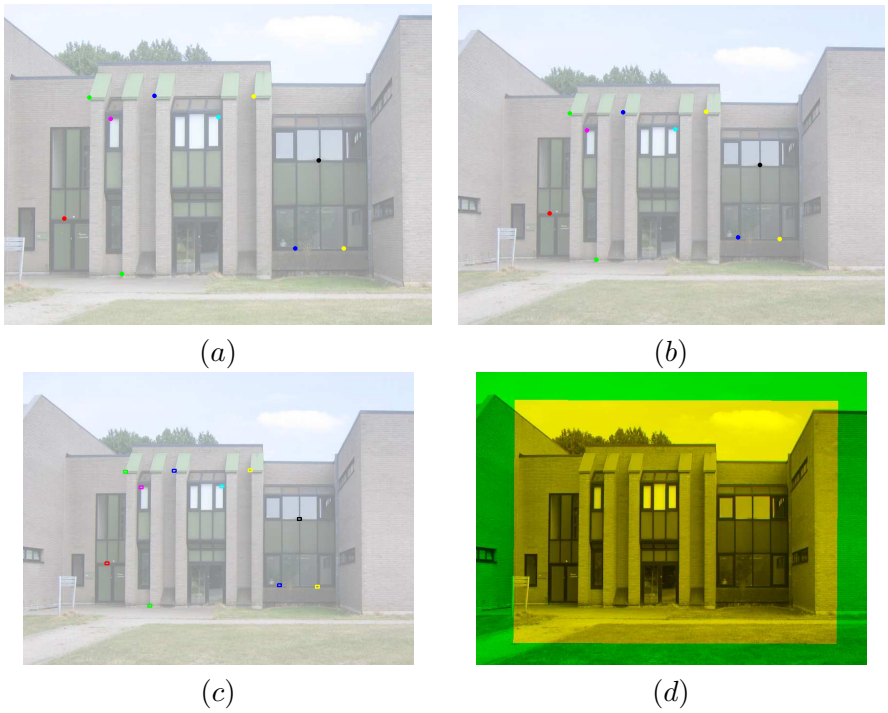


Figure 6.5: Computing the optimal transformation $\mathsf{T}_{\mathsf{opt}}$ for a correspondence set of feature pairs shown in (a) and (b). (c) The regions of minimal size centered on the features in the second image. (d) The registration of the first to the second image with $\mathsf{T}_{\mathsf{opt}}$.

two objects are. The drawings on the right (represented by the feature sets S_2 and S_3) are compared to the leftmost house (S_1). We compute the minimal size of the uncertainty regions (σ_{12}, ρ_{12}) in $\mathcal{R}'_2$ and (σ_{13}, ρ_{13}) in $\mathcal{R}'_3$ for which both $\mathcal{T}(S_1, \mathcal{R}'_2)$ and $\mathcal{T}(S_1, \mathcal{R}'_3)$ are non-empty, as explained in section 6.3.2. The results show that the first two houses (S_1, S_2) are more alike than the first and the third (S_1, S_3).

The similarity measure is not symmetric (remember the discussion about polytope duality in section 5.4.3). As an illustration we compute (σ_{21}, ρ_{21}) for a transformation from S_2 to $\mathcal{R}'_1$. Figure 6.6 (b) shows that the minimal region size for which there still is (at least) one transformation in the TUP $\mathcal{T}(S_2, \mathcal{R}'_1)$ differs from that obtained in Figure 6.6 (a).

This procedure was applied to recognize road marks on the road surface [Maertens, 2006] by comparing them to a database of shapes, as

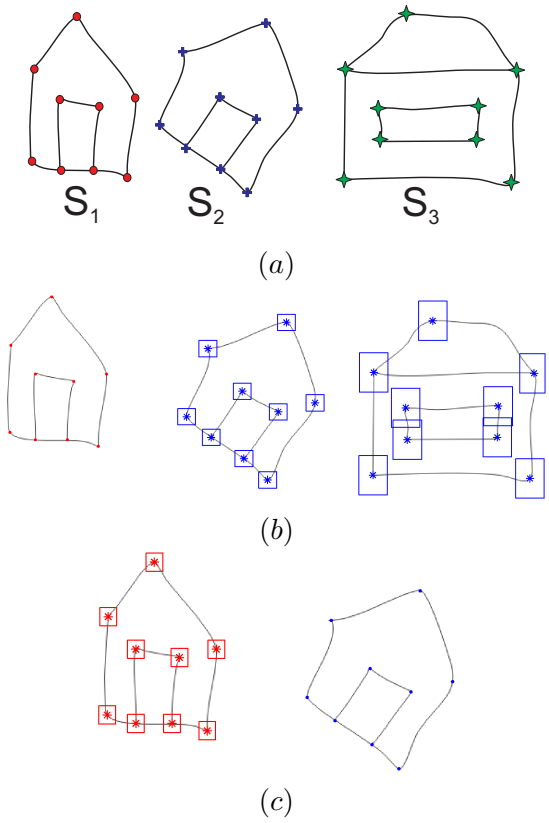


Figure 6.6: (a) How similar are hand drawn sketches? (b) The first drawn house is compared to both other examples by means of the similarity measure. The uncertainty regions in $\mathcal{R}'_2$ must be of size $(\sigma_{12}, \rho_{12}) = (21.71, 19.88)$, so that there is still at least one transformation left in the set $\mathcal{T}(S_1, \mathcal{R}'_2)$. The size of the regions in $\mathcal{R}'_3$ is $(\sigma_{13}, \rho_{13}) = (34.88, 47.85)$. (c) The similarity measure is not symmetric. When transforming S_2 to S_1, the minimal region size for which there still is one transformation in $\mathcal{T}$ is $(\sigma_{21}, \rho_{21}) = (17.97, 19.74)$, which differs from (σ_{12}, ρ_{12}).

illustrated in Figure 6.7. The point sets on the shapes are obtained as the vertices of a (crude) polygonal approximation of the shape. Most feature sets contain distorted feature locations as many road mark shapes are imperfectly segmented due to varying lighting conditions, shadow casted by houses or moving trees, occlusion by other vehicles, etc. The correspondence map for a detected shape to each database shape is obtained through the U-RANSAC approach. Then, the minimal size of the regions for the optimal transformation is computed and applied

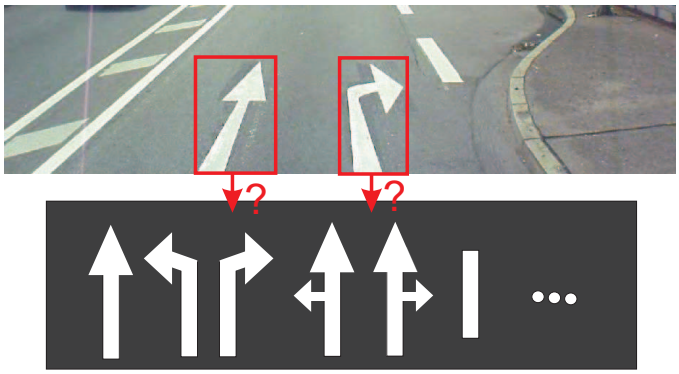


Figure 6.7: The matching procedure in a road marks classification application.

as a similarity measure. The most similar shape is retained. The experimental results support the use and robustness of our method in this application. 95% of the road marks in a video sequence obtained from a moving vehicle were detected and classified correctly using our method [Maertens, 2006]. This result is comparable to the recognition rate of $85 - 97\%$ obtained by some recently published methods [Noda et al., 2009; Kheyrollahi and Breckon, 2010]. A more elaborate evaluation will be necessary to validate our results on sequences in distinct weather conditions, for a wider range of driving speeds, etc.

6.4 Conclusion

In this chapter, we presented the method for computing the *optimal* transformation in the TUP. The optimal transformation is that for which the correspondences can be mapped into uncertainty regions of minimal size, hereby minimizing spatial uncertainty. The concept of TUPs allows to easily define higher-level geometric properties, like the similarity measure, which can still be verified in reasonable time.

We have shown how to fit our uncertainty framework in a standard RANSAC procedure. Although our process may seem more complex than a standard RANSAC approach, for applications where correct samples are inaccurately localized, the U-RANSAC procedure can be effective. It is beneficial to take the positional uncertainty of features into account, hereby reducing the number of required iterations to find a consensus set of sufficient size.

On the other hand, RANSAC-based procedures can be computa-

tionally quite complex, and require the prior estimation of several parameters and the adequate definition of a stopping criterion. To avoid such choices, we will present a method in Chapter 8 to automatically determine a correspondence set from small, and possibly corrupted feature sets without setting stop criteria. We require spatial consistency for pairs in the consensus set, and verify whether the returned consensus set is sufficiently reliable. Another asset is that our framework naturally allows for the incorporation of prior knowledge about the transformation parameters from application details.

Original contributions. Our contributions regarding the consensus set are described in [Teelen and Veelaert, 2005c]. We presented a comparison to the RANSAC approach in [Veelaert and Teelen, 2006a]. [Teelen and Veelaert, 2004a] gives more details concerning the similarity measure.

Chapter 7

Uncertainty of Projective Transformations

Most of the theory developed for the uncertainty framework can be extended toward other, more complex transformation types. This chapter considers the uncertainty of projective transformations or homographies, both for point and line features.

7.1 Introduction

To find a solution for the transformation problems in the case of projective point transformations, we must extend the uncertainty framework that was developed for affine transformations. It is not difficult to restate all propositions for projective transformations. For most propositions the proof is given along the same lines as for the affine transformation. However, some difficulties arise. First, points are written in homogeneous coordinates $\mathbf{x} = (x, y, w)^T$, and second, a homography of the form

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}, \tag{7.1}$$

is only defined up to a scale factor.

As the transformation matrix H has 8 degrees of freedom, $\mathsf{H}(S, S')$ can be computed from a set of at least 4 point correspondences in general position [Hartley and Zisserman, 2003].

7.2 Projective Transformation Uncertainty

Any point $\gamma(x, y, w)^T, \gamma \neq 0$ represents the same point $\mathbf{x}$ in the $\mathbb{R}^2$ image space. If the PUR for a feature point $\mathbf{x'_i} = (x'_i, y'_i, w'_i)^T$ is defined as a convex polygon R'_i in the image space, then the localization uncertainty for that point can be represented as a pyramid C'_i minus the origin in the $\mathbb{R}^3$ parameter space $x'y'w'$. For a n-polygonal PUR in the image space, the positional uncertainty pyramid (PUP) C'_i is delimited by n halfplanes,

$$r_i x' + s_i y' + t_i w' \leq 0, 1 \leq i \leq n. \tag{7.2}$$

Figure 7.1 (a) illustrates the square PURs R'_i around the correspondences $\mathbf{x'_i}$ for each $\mathbf{x_i}$. The corresponding PUPs C'_i are given in (b).

7.2.1 Transformation Uncertainty Representation

When given at least 4 correspondence pairs of points $\mathbf{x_i}$ and polygonal PURs R'_i (or PUPs C'_i) in general position, we can represent the transformation uncertainty as a convex TUP $\mathcal{H}$ in a 9-dimensional parameter space $h_1, \ldots, h_9$. $\mathcal{H}(S, \mathcal{R}')$ is computed as the convex hull of all transformations that map the points $\mathbf{x_i} \in S$ within their corresponding uncertainty polygons $R'_i \in \mathcal{R}'$. $\mathcal{H}$ can be represented as a set of inequalities, that is obtained by substitution of the equations (7.1) into the inequalities (7.2) of the PUPs.

The TUP can also be given in a dual representation, as the closed convex span of the finite set of polytope vertices. This result is supported by the following propositions for projective transformations, stated in analogy with the propositions 5.2 and 5.3 for affine transformations [Teelen and Veelaert, 2009].

Proposition 7.1 *The projective transformation of a convex combination of points is equal to the convex combination of the transformed points,* $\mathsf{H}(\alpha_1 \mathbf{x_1} + \alpha_2 \mathbf{x_2}) = \alpha_1 \mathsf{H}(\mathbf{x_1}) + \alpha_2 \mathsf{H}(\mathbf{x_2})$*, provided* $\alpha_1 + \alpha_2 = 1$ *with* $0 \leq \alpha_i \leq 1$*.*

Proposition 7.2 *The application of a convex combination of projective transformations to a point* $\mathbf{x_i}$ *is equal to the convex combination of the transformed points,* $(\alpha_1 \mathsf{H}_1 + \alpha_2 \mathsf{H}_2)(\mathbf{x}) = \alpha_1 \mathsf{H}_1(\mathbf{x}) + \alpha_2 \mathsf{H}_2(\mathbf{x})$*, with* $\alpha_1 + \alpha_2 = 1$ *and* $0 \leq \alpha_i \leq 1$*.*

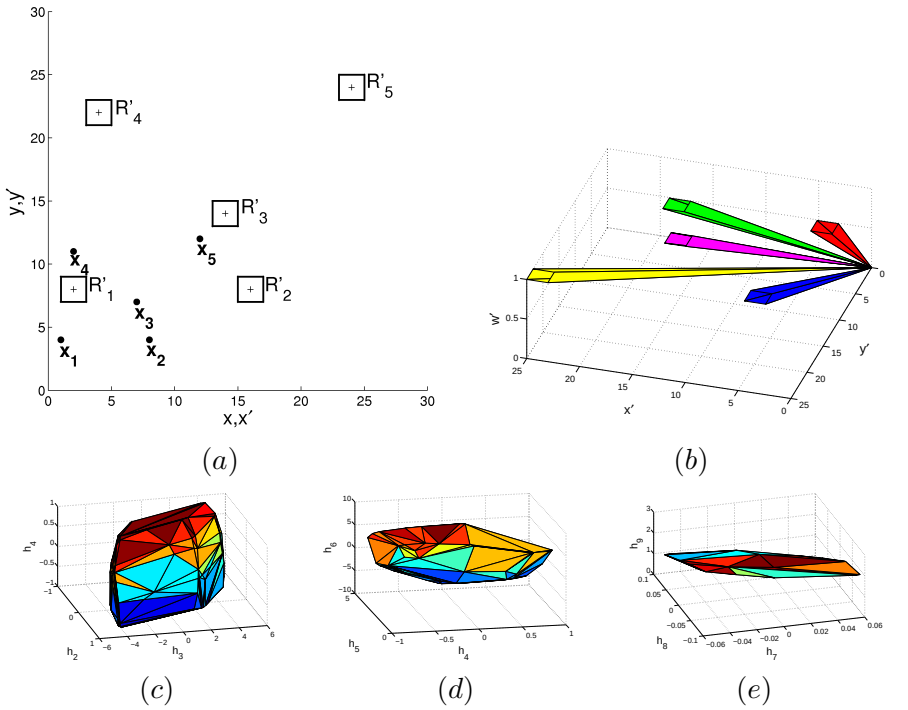


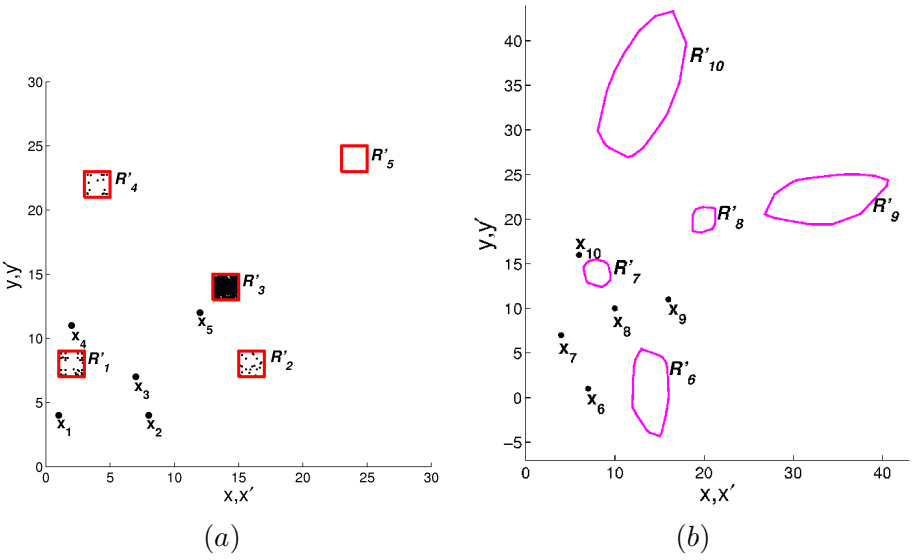
Figure 7.1: (a) The points $\mathbf{x_i}$ and their corresponding square PURs R'_i in the image space. (b) The PUPs C'_i constructed for each R'_i. $(c-e)$ The transformation uncertainty is represented by the TUP $\mathcal{H}$ in 9D (here projected to a 3D parameter space in resp. $(c), (d), (e)$).

Both propositions allow us to write each transformation H in the TUP $\mathcal{H}$ as a convex combination of the vertices of $\mathcal{H}$.

When polygonal PURs R'_i in the second image are given as a correspondence for each feature point $\mathbf{x_i}$ in the first image, a TUP in the 9-dimensional transformation space can be computed. We compute the TUP $\mathcal{H}(S, \mathcal{R}')$, which considers the computation of each $\mathsf{H}(S, S')$ as a subproblem. This TUP is illustrated for the situation in Figure 7.1 (a) by the projections of the TUP to 3D subspaces in $(c-e)$.

7.2.2 Implied Uncertainty Regions

The computation of RoIs for transformed features requires a solution for both $\mathcal{H}(S, \mathcal{R}')$ and $\mathcal{R}'(S, \mathcal{H})$. To find the implied regions, we can restate proposition 5.5 for homographies, since proposition 7.2 holds. The proof is immediate [Teelen and Veelaert, 2009], as each transformation



(a) (b)

Figure 7.2: (a) The vertices of the TUP $\mathcal{H}$ in Figure 7.1 are used to project the points $\mathbf{x_i}$ to their IURs. Notice that $R'(\mathbf{x_i}, \mathcal{H}) \subset R'_i$. (b) A set of other points $\mathbf{x_j}$ is projected to their IURs with the same TUP $\mathcal{H}$. Notice the irregular shape of the IURs.

H belongs to the convex span of the vertices V_i of the TUP $\mathcal{H}$.

Proposition 7.3 *Let* $\mathbf{x}$ *be a point, and let* $\mathcal{H}$ *be a given convex polytope of projective transformations. Let* V_i *denote the transformations that correspond to the vertices of the polytope* $\mathcal{H}$*. Then* $R'(\mathbf{x}, \mathcal{H})$ *is the convex hull of the points* $\mathbf{x'_i}$*, where* $\mathbf{x'_i} = \mathsf{V}_i(\mathbf{x})$.

Once the vertices V_i of a TUP $\mathcal{H}(S, \mathcal{R}')$ are computed (as in Figure 7.1 $(c-e)$), they can be used to compute the IURs for a set of other points $\mathbf{x_j} \in S_o$ in the first image. Figure 7.2 (b) illustrates the IURs $\mathcal{R}'_o(S_o, \mathcal{H})$. Notice their irregular shape.

The area of the IURs in $\mathcal{R}'_o(S_o, \mathcal{H})$ is equally large as that of the regions R'_i, when the $\mathbf{x_j}$ are located within the convex hull of all $\mathbf{x_i}$. The area of the IURs grows quickly, when the $\mathbf{x_j}$ are located further away. Therefore, it is vitally important to adequately choose the sample subset of correspondences from which the TUP is computed in a U-RANSAC like approach, i.e., the sample points must be in a non-degenerate configuration, well spread over the image, and not lie too close to each other.

7.2.3 Examples

We will now illustrate some example applications where the computation of RoIs for a projective TUP can play an important role.

Computing RoIs for Tracking Purposes

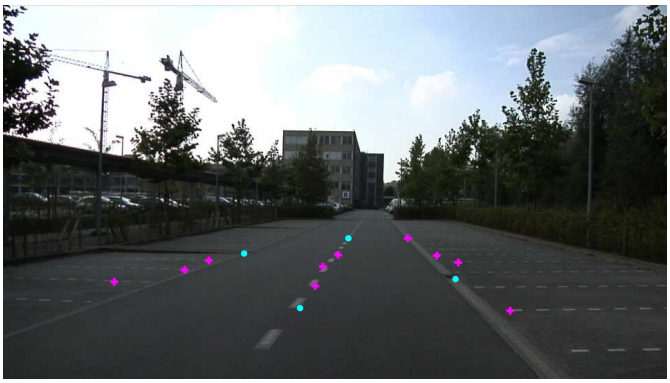
Figure 7.3 shows two images of a sequence recorded from a moving vehicle in a traffic situation. We consider the road as one ground plane in the 3D world. In this example, the projective transformation will allow us to accurately determine the mapping of geometric features to the same road plane in different images.

A projective TUP is computed for a sample set of 4 point-region correspondence pairs, indicated in Figure 7.3 $(a-b)$ by full dots and square regions. Then the RoIs for a set S_o of other points (indicated by crosses (a)) are obtained as the IURs $\mathcal{R}'(S_o, \mathcal{H})$ (b). Again, the IURs are of an irregular shape, due to the perspective effects introduced by the projective transformation. We notice that the regions in between those of the initial subset $\mathcal{R}'$ are accurate, but that the area of the IURs increases fast for points lying farther away from those in the sample subset.

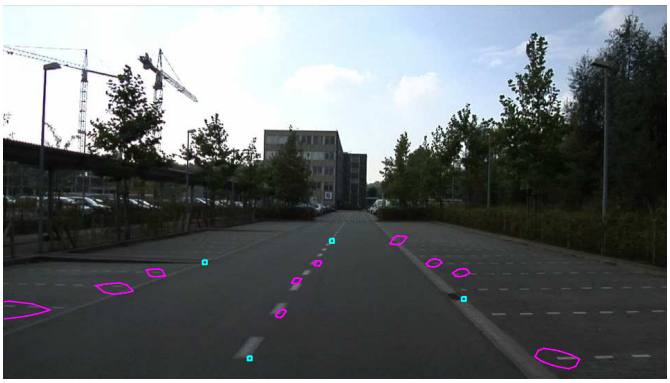
Computing RoIs in Image Stitching Applications

Most stitching algorithms consider a large set of feature descriptors in both images to be able to estimate the registration transformation as accurately as possible. In the common approach, all n features in the left image are compared to all m features in the right image based on appearance, i.e. by computing a distance metric for their descriptors. This procedure requires $n \times m$ distance computations. The closest correspondence is compared to the second, and accepted if they are sufficiently far apart.

We first compute an initial estimate for the TUP based on a few prominent correspondences, for which the similarity measure is sufficiently large (like in the PROSAC approach). Then we try to refine our estimate by computing the RoIs for a large set of other features. Figure 7.4 (a) shows the resulting correspondence set from a U-RANSAC approach for a limited set of pronounced SIFT features. The actual correspondences are to be found among the possible matches for each feature in a bounded RoI of limited size based on appearance. For the



(a)



(b)

Figure 7.3: The corresponding pairs $(\mathbf{x_i}, R'_i)$ are indicated on frame t_0 and t_{25} of a sequence obtained from a moving vehicle (by dots in (a), and squares in (b)). The TUP $\mathcal{H}(S, \mathcal{R}')$ is computed for the indicated correspondences. For a set of other features (indicated by crosses (a)), the RoIs are obtained as the IURs (polygonal regions in (b)).

larger set of the detected SIFT features in (b), the RoIs are computed and shown in (c). The (coarse) demand for spatial consistency reduces the number of discarded correspondences when compared to the common procedure. In this example, we establish almost 25% more correct correspondences than obtained by the common approach.

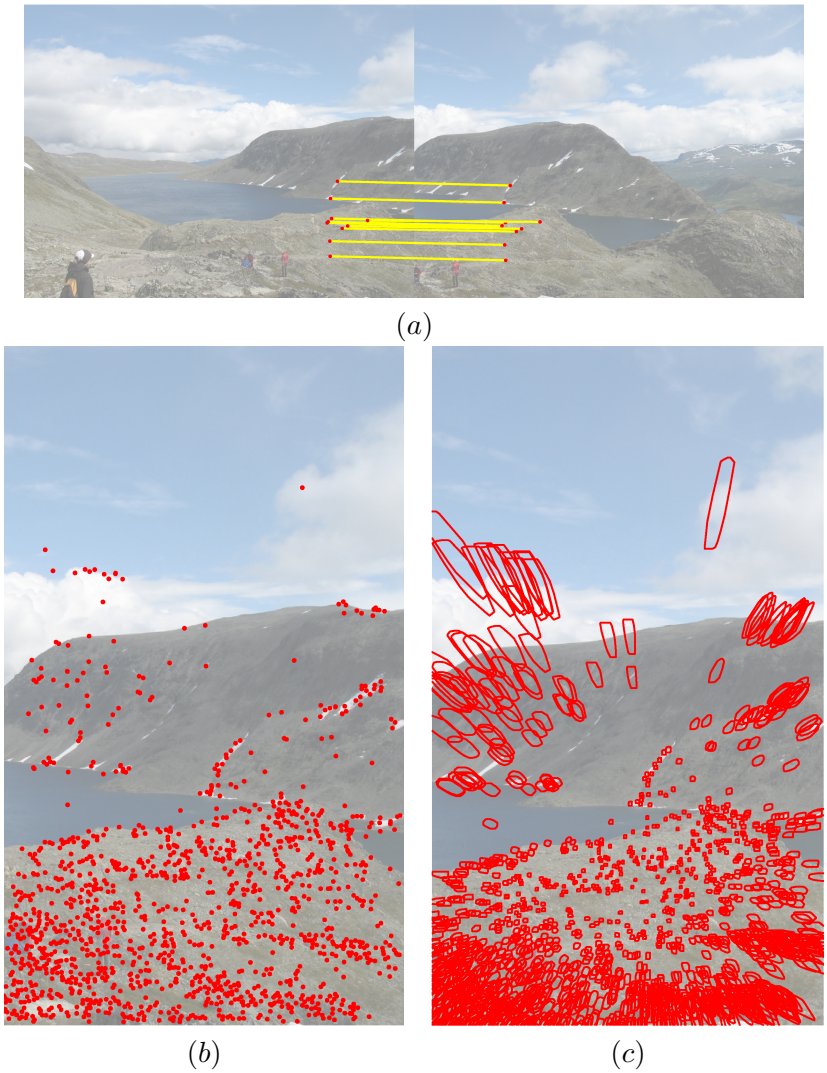


Figure 7.4: Determining reliable RoIs in a stitching application. (a) The result of applying the U-RANSAC approach for a small subset of SIFT features in each image. (b) The set of detected SIFT features. (c) The RoI for each of the indicated correspondences in (b).

7.3 Transformation Uncertainty for Straight Lines

If the projective transformation H for points (Eq. 7.1) is non-singular, the line $\mathbf{l} = (p, q, r)^T$ is transformed to a line $\mathbf{l}' = (p', q', r')^T$ by a linear transformation

$$\mathbf{l}' = \mathsf{G}\mathbf{l} = |\mathsf{H}|\mathsf{H}^{-T}\mathbf{l}. \tag{7.3}$$

Then we can state the following property for projective transformations in analogy with Proposition 5.7 for the affine transformation [Teelen and Veelaert, 2009].

Proposition 7.4 *Let* H *be a projective transformation of the form (7.1). The transformation of a convex combination of line parameters (for which* $\alpha_1 + \alpha_2 = 1$*, and* $0 \leq \alpha_1, \alpha_2 \leq 1$*) is equal to the convex combination of the transformed line parameters. That is,* $\mathsf{H} < \alpha_1\mathbf{l_1} + \alpha_2\mathbf{l_2} >= \alpha_1\mathsf{H} < \mathbf{l_1} > +\alpha_2\mathsf{H} < \mathbf{l_2} >$.

When considering the homography G (7.3) for straight lines, one can proceed as done for the points, as the description of the localization uncertainty for lines and points is similar in both parameter spaces.

We give a simple example of how to solve the RoI-problem for uncertainty on the location of lines for projective transformations in Figure 7.5. A solution for the RoI problem requires the computation of both the TUP from a set of line correspondences, i.e. $\mathcal{G}(S, \mathcal{R}')$, and the IURs of that TUP for a set S_o of other lines, i.e. $\mathcal{R}'_o(S_o, \mathcal{G})$.

We compute the TUP from the known correspondences $(\mathbf{l_i}, \mathbf{l'_i})$, indicated as the solid lines $\mathbf{l_i} \in S$ in Figure 7.5 (a) and the corresponding PURs in $\mathcal{R}'$ given by the dashed lines around the solid lines $\mathbf{l'_i}$ in (b). The PUPs in the lines parameter domain (p', q', r') are defined as in section 5.5.1. The uncertainty of the homography is represented by the TUP $\mathcal{G}(S, \mathcal{R})$ in the 9D parameter space $g_1, \ldots, g_9$ (i.e. the coefficients of G in Eq. 7.3). Figure 7.2 (c) shows a projection of that TUP onto a 3D parameter space $g_1g_2g_3$.

The IURs for a set of other lines $\mathbf{l_j} \in S_o$ (the dashed lines in (a)) are computed with (the vertices of) $\mathcal{G}(S, \mathcal{R})$. These regions are shown in the image domain (d): each dashed line represents a map of one $\mathbf{l_j}$ by a transformation V_i that is a vertex of $\mathcal{G}(S, \mathcal{R})$.

If the uncertainty on the lines position is larger (as indicated in (e)), then also the uncertainty on the location of the transformed lines will increase (f). Here the uncertainty of straight line transformations is

treated similarly as that of point transformations. However, we want to consider the point and line transformation in one common parameter space $h_1, \ldots, h_9$.

7.3.1 Line and Point Transformations in a Common Parameter Space

If we consider the transformation uncertainty in one parameter space $h_1, \ldots, h_9$ for both point *and* line features, then some difficulties arise, as, unfortunately, the transformed line parameters do not form a polytope in the general case. The equations for the transformed straight line parameters can be deducted from (7.3), i.e.

$$\begin{array}{rcl} p' & = & -h_6h_8p + h_5h_9p + h_6h_7q - h_4h_9q - h_5h_7r + h_4h_8r \\ q' & = & h_3h_8p - h_2h_9p - h_3h_7q + h_1h_9q + h_2h_7r - h_1h_8r \\ r' & = & -h_3h_5p + h_2h_6p + h_3h_4q - h_1h_6q - h_2h_4r + h_1h_5r \end{array} \quad . \tag{7.4}$$

These are not linear expressions in the parameters h_i. We also encountered this situation for affine transformations.

There are two solutions possible. Either we try to find more constrained transformations so that the non-linearity is resolved, or we approximate the parameter sets by convex polytopes.

Constrained Transformations

The vertices of the conflict graph in Figure 7.6 each represent one parameter h_i. Every two parameter vertices are connected by an edge when these occur together in one term in the expressions (7.4). Notice that the conflict graph for the projective transformation parameters in Figure 7.6 is a generalization of that for the affine transformed line parameters in Figure 5.22. That is, the graph in Figure 5.22 is a subgraph of the conflict graph in Figure 7.6.

Although the equations (7.4) are nonlinear in the parameter domain h_i, we can select a subset of parameters that are not mutually connected in the conflict graph. In these cases, the equations in (7.4) can be linearized, by the vanishing or fixing of the parameters not in the selected subset. Then, the transformations are more constrained, such as

$$\begin{bmatrix} h_1 & 0 & h_3 \\ 0 & h_1 & h_6 \\ 0 & 0 & h_9 \end{bmatrix} \tag{7.5}$$

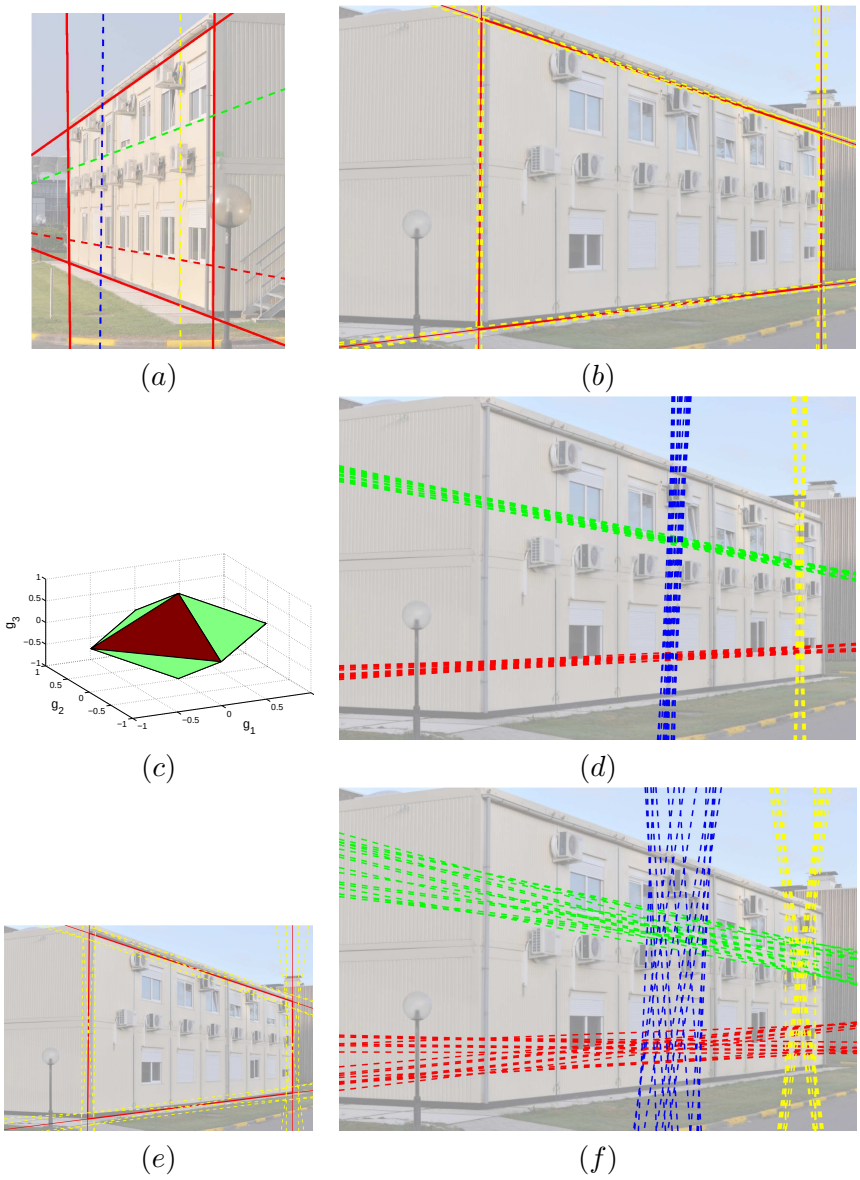


Figure 7.5: Region of Interest computation for projective transformations of lines. $(a-b)$ The corresponding lines are indicated in full, while the dashed lines represent the localization uncertainty as PURs in the image space. (c) The transformation uncertainty is represented as a TUP $\mathcal{H}$ in a 9D parameter space (here projected to the 3D parameter space $s_1 s_2 s_3$). (d) The IURs are shown as the projection of each dashed line in (a) by one vertex of the TUP $\mathcal{H}$. $(e-f)$ If the localization uncertainty on the lines position is larger, then also the localization uncertainty of the transformed lines will increase.

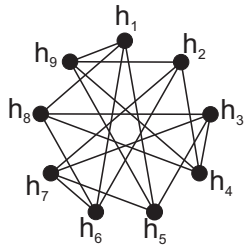


Figure 7.6: Conflict graph for the parameters in the non-linear expressions of Eq. 7.4.

for which the expressions of Eq. 7.4 can be rewritten as $(p', q', r') = (h_9 p, h_9 q, -h_3 p - h_6 q + h_1 r)$. This special case is a generalization of the constrained affine transformation A_{1136} (5.12). However, the parameters h_7, h_8 that must account for the perspective effects are also fixed to 0. Therefore, if we want to consider a general projective transformation, we must solve the *unconstrained* case, where no parameters are vanishing.

Approximating Parameter Sets by Convex Polytopes

We can again generalize the results obtained for the affine transformations. Also the line parameters transformed by the convex span of two homographies must lie on a conic segment.

Proposition 7.5 *Let* $\mathsf{H}_1, \mathsf{H}_2$ *be two homographies of the form (7.1), and let* $\mathsf{H} = t\mathsf{H}_1 + (1-t)\mathsf{H}_2$, *with* $0 \leq t \leq 1$, *denote the transformations in their convex span. Let* α, β *be the parameters of the line* $y = \alpha x + \beta$. *Then the parameters* α', β' *of the line transformed by* H *lie on a common conic, for* $0 \leq t \leq 1$.

To obtain a convex polygonal uncertainty region for the line parameters α, β, the two solutions presented earlier in section 5.5.3 can be applied, i.e. choosing intermediate vertices on the conic segments, or computing the convex hull of the enclosing quadrangles of the conic segments.

Figure 7.7 shows an example for the line $y = x/3 - 5/4$ with parameter vector $\mathbf{l} = (1/3, -1, -5/4)$, transformed by a convex set of transformations. Each transformation H belongs to the convex span $\mathsf{H} = \sum_i \gamma_i \mathsf{H}_i$ with coefficients $(h_1, \ldots, h_9) = \gamma_1(2, 2, 3; 4, 4, 1; 2, 7, 1) +$

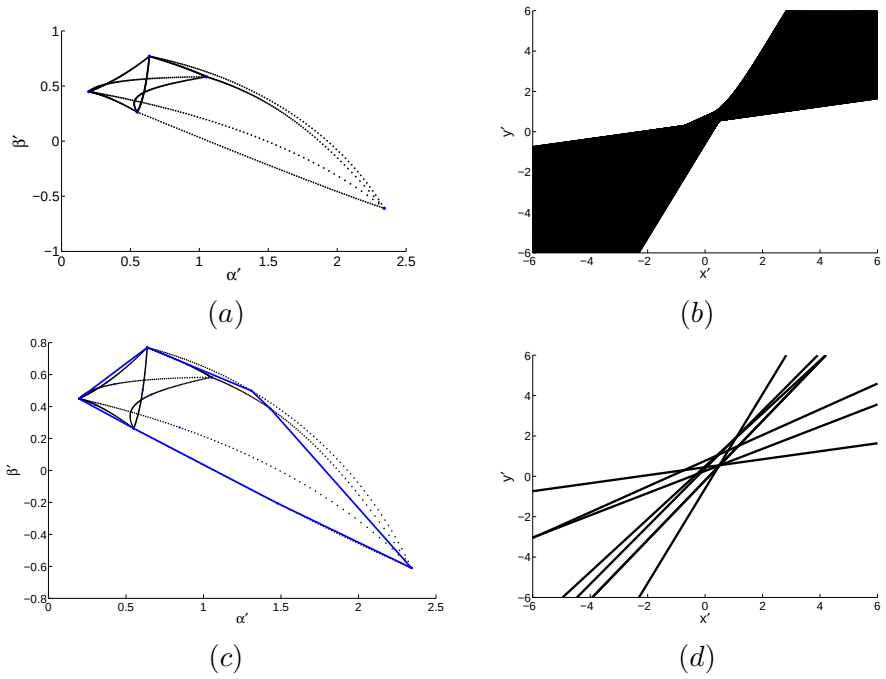


Figure 7.7: (a) The transformed line parameter set after transforming the line $\mathbf{l} = [1/3, -1, -5/4]^T$ with a given TUP. (b) The transformed lines in the image domain for each line parameter indicated in (a). (c) The convex hull of a selection of parameter points on the conic segments (solid lines). (d) The transformed lines corresponding to the vertices of the convex hull in (c).

$\gamma_2(2,1,3;4,4,4;4,3,1) + \gamma_3(1,3,4;3,2,4;1,2,1) + \gamma_4(3,2,2;2,4,4;4,3,1) + \gamma_5(4,2,4;4,2,1;7,4,1)$, with $\sum_i \gamma_i = 1$ and $0 \leq \gamma_i \leq 1$. Figure 7.7 (a) shows the range of the parameters (α', β') of the transformed line $y = \alpha' x + \beta'$. The conic sections correspond to parameters transformed by convex spans of the form $\gamma_i \mathsf{H}_i + \gamma_j \mathsf{H}_j$, $1 \leq i < j \leq 5$, thus the boundaries of the convex span are included. The transformed line parameters are shown in the image domain in Figure 7.7 (b).

We can now approximate the parameter set by taking the convex hull of the transformed line parameters $\mathsf{H}_i < \mathbf{l} >$, as well as $((\mathsf{H}_i + \mathsf{H}_j)/2) < \mathbf{l} >$. The convex approximation is shown in solid lines in Figure 7.7 (c). Notice that this approach produces a good, but not necessarily an enclosing approximation for the set of transformed line parameters. If we need an enclosing set, we can proceed as in 5.5.3 for the enclosing quadrangles.

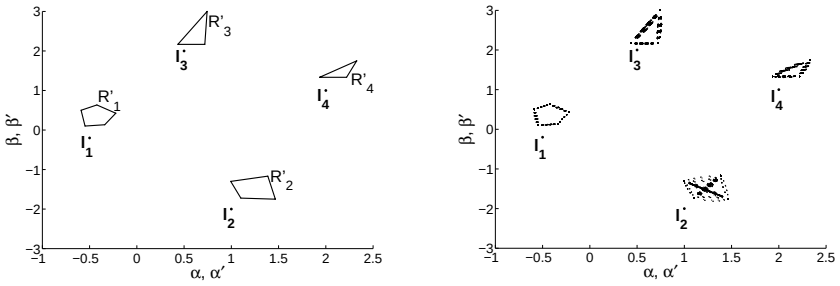
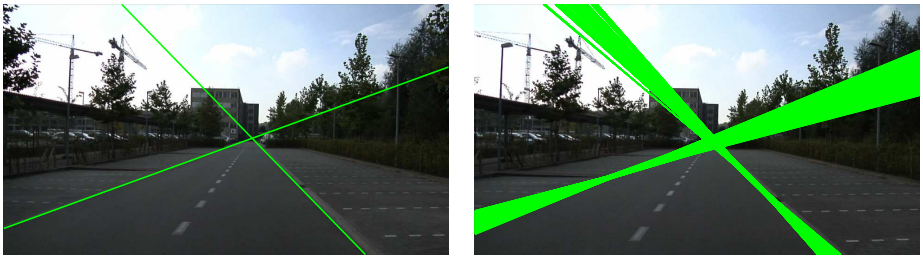


Figure 7.8: Approximating transformation parameter sets in projective transformation uncertainty problems for straight lines in a common parameter space $h_1 \dots h_9$. (a) 4 lines $\mathbf{l_i}$ and corresponding PURs R'_i in the line parameter space $\alpha'\beta'$. From the correspondences, we compute an approximation $\tilde{W}$ for the transformation uncertainty set. (b) The transformed line parameters (on conic segments), after transformation by the vertices of the set $\tilde{W}$.

We can also consider the computation of a transformation uncertainty set for the mapping of a set of lines $\mathbf{l_i}$ onto their corresponding PURs of line parameters, as illustrated in Figure 7.8 (a). Suppose we want to map 4 lines $\mathbf{l_i}$ and their corresponding uncertainty polygons R'_i in the line parameter space $\alpha'\beta'$, we want to compute the transformations H that map each line $\mathbf{l_i}$ onto a line whose parameters lie in the polygon R'_i. When mapping the line parameters of a line $\mathbf{l_i}$ on the set of parameters from a convex combination of line parameters, the resulting set of transformations W is not a convex combination of transformations. However, we can approximate W by a set of transformations $\tilde{W}$, that is the convex hull of all transformations H_i, mapping the lines $\mathbf{l_i}$ onto 4 distinct vertices of the corresponding R'_i. If we map the lines $\mathbf{l_i}$ by the transformations in this set, we obtain transformed parameters on conic segments as shown in Figure 7.8 (b). If the transformed parameters lie close to the R'_i, the transformation set $\tilde{W}$ is a good approximation for W. This procedure appears to result in sufficiently accurate approximations for transformation sets.

7.3.2 Examples

We give some examples to illustrate the computation of RoIs of any polygonal shape for line features under projective transformation.



(a) (b)

Figure 7.9: (a) For each detected line feature, we can compute a RoI in another frame with the TUP computed for the situation indicated in Figure 7.3. (b) The IURs for the lines in the image space.

Computing RoIs for Tracking Purposes

We consider the example of observing a traffic situation with a camera mounted in a vehicle. By tracking the lines that constitute road marks, it is possible to predict where the driving lane is, or direct a driver to the correct exit lane at crossroads, possibly in real time when using an appropriate platform [Coppens and Van Severen, 2008; De Gols and Pauwels, 2009].

Section 7.2.3 presented an example of computing the RoIs for points on the road surface. This can easily be extended to line features. Some of the most prominent point-region correspondence pairs were related by a TUP $\mathcal{H}$ in Figure 7.3 $(a-b)$. Other correspondences, both points and lines, can be found in RoIs as IURs for those correspondences. Figure 7.9 (b) shows the projection with each vertex of $\mathcal{H}$ of the straight lines $\mathbf{l_i}$ that are detected as both sides of the road. Candidate matching lines for each $\mathbf{l_i}$ are discriminated as those lines whose parameters are located in the corresponding PUP C_i' in the parameter space.

RoIs in Rectification Examples

The analysis of the performance of soccer players during matches based on multi-camera video images involves knowledge about the instantaneous position of each player on the field. From the positional information, we can derive statistics such as the total distance covered by a player during a time period, the intensity zones, or conduct a tactical analysis. Therefore, it is required to accurately estimate the mapping of one view on a soccer field onto the actual 2D dimensions of that field, as

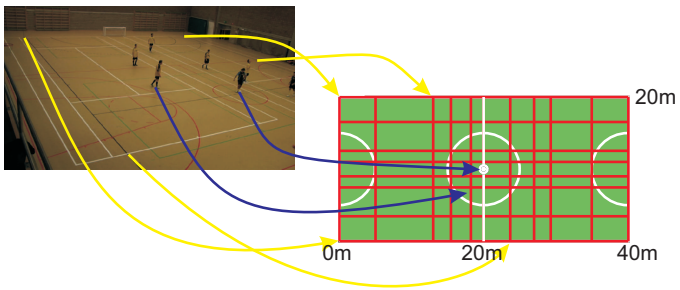


Figure 7.10: Rectification of a view on the soccer field: find the projection of the image to the 2D field world coordinates, or vice versa.

illustrated in Figure 7.10. We can then use that homography to project the trajectories of each player on the real world field model.

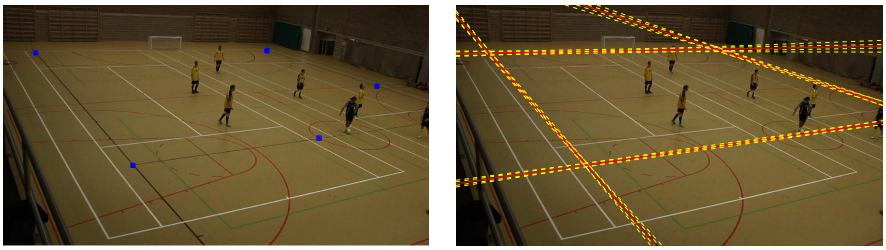
It is not the image as such that must be rectified, instead, we want to establish a reliable correspondence map from the field in the image to a 2D model of the field. The world coordinates of a set of lines on the field (indicated by the overlaid lines in Figure 7.10) are accurately known in advance, because we can measure them on the field.

We could let a user quickly indicate the endpoints of 4 line segments in the image as a correspondence for lines with known positions on the field, as illustrated in Figure 7.11 (a). We assume that we will most likely only get a rough estimation of the actual position, and that there remains some uncertainty about the position of the lines (b).

Even more, computing the homography from a minimal set of 4 correspondences is not that robust. This can be verified by computing the condition number of the linear equations involved in the computations. The condition number measures how numerically well-conditioned a problem is, i.e. how sensitive is the solution of a system of linear equations to errors in the data. It gives an indication of the accuracy of the results from matrix inversion and the linear equation solution.

For example, if we use the correspondences of the 4 lines shown in full in 7.11 (b), we obtain a set of linear equations for which the condition number is in the range of 10^{18}, where values near to 1 indicate a well conditioned matrix. We expect that a more reliable result can be obtained from a larger set of correspondences. We can also take the uncertainty on the location of the lines into account, as represented in the image space by the dashed lines in Figure 7.11(b).

A set of candidate matching lines is detected, as shown in Figure



(a) (b)

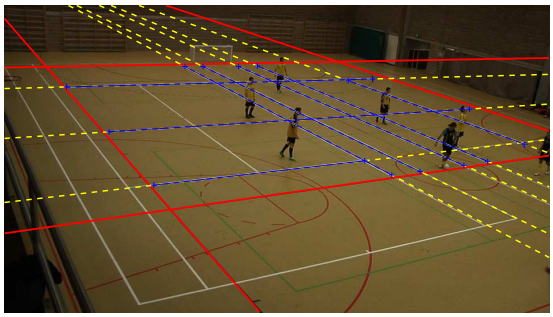
Figure 7.11: Using RoIs for straight lines in the rectification of soccer field data. (a) The user indicated endpoints for the line segments. (b) The corresponding uncertainty regions for the lines superimposed in the image domain.

7.12. Now, we apply a U-RANSAC approach to determine a sufficiently large consensus set, while the positional uncertainty on the detected lines is taken into account. Figure 7.12 $(a-b)$ shows an example of a well chosen sample set, indicated in solid lines. The TUP is computed from the indicated line correspondences, and used to find the implied PUPs for the other known field lines. We can then determine the consensus set by checking whether the parameters for the detected lines are located in the uncertainty pyramids $(c-d)$.

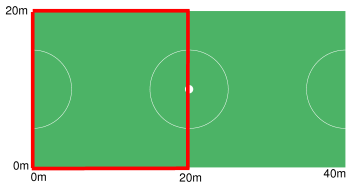
From the larger set of correspondences, we obtain a more robust estimate H_{est} for the rectifying homography than the estimate H_{ini} for the 4 manually indicated correspondences. In this example, the condition number decreases significantly, by a factor 10^{12}, when computed for the system of 12 equations instead of 4. The effect is immediate: when mapping points from the image space onto the 2D field, their position is more accurately localized. We compute the intersection points $\mathbf{p_i}$ of each line pair in the image, and compute both the projection $\mathbf{p}_{est} = H_{est}\mathbf{p_i}$ and $\mathbf{p}_{ini} = H_{ini}\mathbf{p_i}$. On average the $\mathbf{p}_{est}$ are closer to the expected field position than the $\mathbf{p}_{ini}$, as indicated in Figure 7.13.

The estimated homography can then be applied to compute the actual trajectories for each player in real world coordinates. However, if the position of each player in the frame can only be estimated up to some uncertainty denoted by the image regions in Figure 7.14(a), the location is mapped to an uncertainty polygon $R'(R, H_{est})$ on the field model in (b). If multiple cameras are available, we can estimate the position of the players more accurately, as illustrated in Figure 7.14$(c-d)$

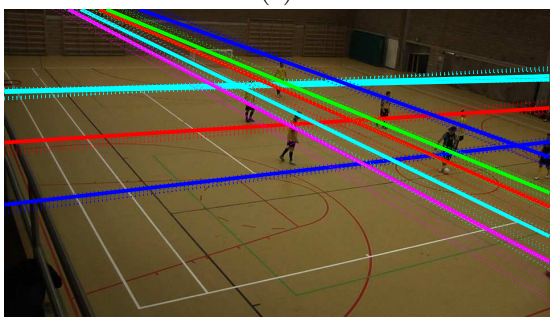
A possible extension of this method lies in its application for sequences captured with dynamic cameras. Then the homography must



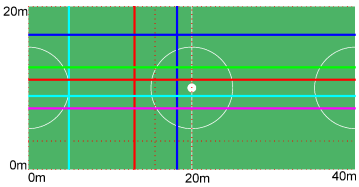
(a)



(b)

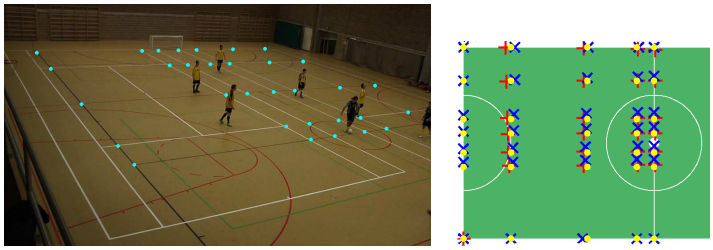


(c)



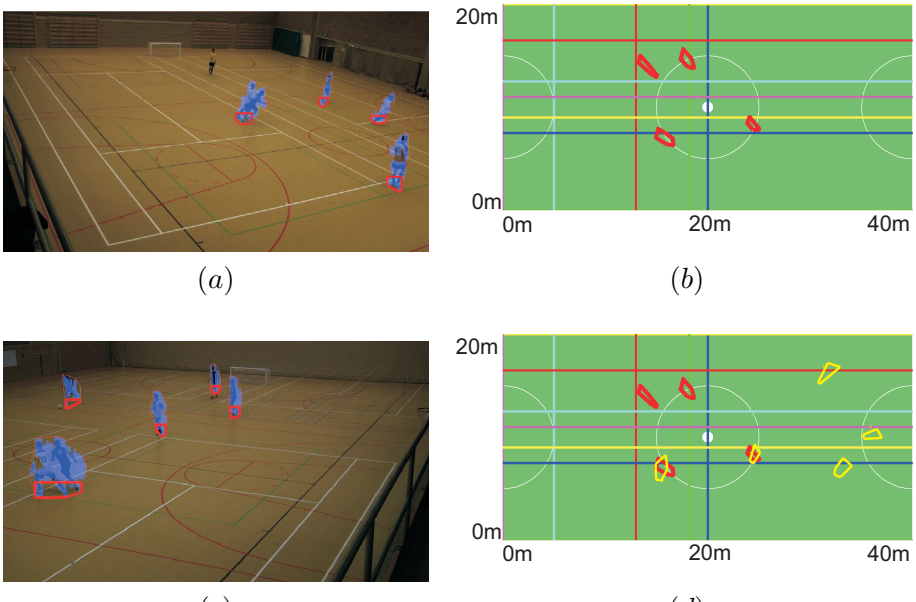
(d)

Figure 7.12: (a) A sample subset S_i' of 4 lines is selected from the lines detected by RANSAC in the Canny edge map. (b) The corresponding 4 lines in S on the 2D field. We compute a TUP for the 4 selected correspondence pairs with PUPs for each of selected lines in S_i'. (c) An implied PUP C_i' is computed for all lines $\mathbf{l_i}$ in S, and the associated IURs are shown as dashed lines. If a line $\mathbf{l_j'}$ is detected in C_i', we add the correspondence pair $(\mathbf{l_i}, \mathbf{l_j'})$ to the consensus set of this TUP. (d) The actual correspondences for the solid lines in (c) are given by matching colors.



(a) (b)

Figure 7.13: (a) The intersection points $\mathbf{p_i}$ of the detected lines. (b) The image $\mathsf{H}_{est}\mathbf{p_i}$ $(+)$ lies on the average closer to the expected position in the field $(\bullet)$, than the map $\mathsf{H}_{ini}\mathbf{p_i}$ $(\times)$.



(a) (b)

(c) (d)

Figure 7.14: (a) We estimate the position of players as a polygonal region in the image derived from the result of a foreground estimation technique. (b) These polygons are then mapped onto the soccer field as $\mathcal{R}'(\mathcal{R}, H_{est})$. (c) The detected foreground regions in a second camera view. (d) When mapping the detected foreground regions of both views onto the field model, the position of the player in the overlapping view can be computed as the intersection of the mapped regions.

be estimated from frame to frame, so that the players can be accurately positioned on the 2D field model. In each (limited) view on the field by one camera, only a very limited number of line features is visible (certainly for sequences of outdoor soccer). Reliable point feature correspondences are difficult to find in the grass surface, certainly in compressed images. Then it is important to give an initial (possibly coarse) estimate of the transformation based on few feature correspondences, and even more, an indication of how uncertain (or reliable) that estimate is. Thus, our uncertainty framework could prove its use. The next chapter will focus more on reliability measures in the estimation of transformations.

7.4 Conclusion and Future Work

This chapter has shown how to include projective transformations and homogeneous coordinates into our transformation uncertainty framework. It is straightforward to extend the previously stated propositions for more complex transformations. Part of our future work will be to show how our uncertainty framework can be extended toward the computation of uncertainty on the fundamental matrix or the perspective transformation.

An overview of computation methods of the fundamental matrix including uncertainty is given in [Zhang, 1998; Csurka et al., 1997]. In most cases, the uncertainty is modeled by an approximation based on the lower order moments, i.e., mean and covariance. However, the moments estimation is computed for a large set of samples, free from outliers (e.g. on a consensus set or a set of manually selected points). Recent work, such as [Sur et al., 2008], concentrates on the estimation of the fundamental matrix for a sample set of limited size, or with a considerable amount of outliers.

Original contributions. The extension of our uncertainty framework toward projective transformations is mainly described in [Teelen and Veelaert, 2009].

Chapter 8

Consistency and Confidence

The previous chapters illustrated the use of our uncertainty framework in the computation of RoIs in a second image for a set of features in a first image. In this chapter we discuss methods to verify whether correspondence sets are spatially consistent. We also quantify the reliability of these correspondence procedures by introducing reliability measures for the consensus set of a transformation uncertainty polytope.

8.1 Introduction

Up to now the methods in our uncertainty framework were mainly applied to solve the RoI problem, i.e., in which image region can we find possible matches? From the *multiple candidate* matches that are located in each RoI, we must select *one true* correspondence for each feature.

Figure 8.1 illustrates some of the problems occurring in the considered applications, here a tracking task where vehicles are moving in a complex scene observed by a static camera. There are typically few features found on each vehicle, and these cannot be localized robustly in consecutive frames due to the changing illumination on 3D surfaces, noise, discretization effects, etc. In this kind of application we know that the displacement in between two consecutive frames is quite limited, given the relative speed of the vehicle. Thus, we assume that the candidate matches in frame $(t+1)$ must occur in a local neighborhood of the features in frame (t). Typically, for each feature there can be multiple candidate correspondences, i.e., a set of putative correspondences

Figure 8.1: Find the spatially consistent matches for a subset of few features (crosses) in frame (t) among the possible candidate matches (dots) located in a restricted search area in frame $(t+1)$. From this limited amount of data, corrupted by outliers, we must deduce the actual correspondence set (thick lines).

in frame (t). As the background is static, the matching procedure must preferably be able to distinguish (at least) two different transformation models.

This chapter presents our framework for extracting reliable correspondence sets under the described circumstances. Our framework must be able to distinguish more than one consistent correspondence set at a time in a corrupted feature set of limited size, with preferably no parameters to tune. In comparison, a RANSAC procedure expects a rather large amount of input data, with not too many false candidates, and an indication of the expected parameters.

A first advantage of our framework is the easy incorporation of *prior knowledge* about the feature transformations. For instance, the relative displacement of features in between two consecutive frames can be predicted based on a previously estimated transformation, or it can be restricted according to an application specific motion model. Or, if we know the internal and external camera parameters, we can accurately estimate a bounded image region in which to look for candidate matches for each feature. Our approach naturally allows for the inclusion of prior knowledge about transformations.

In this chapter, we will introduce *reliability measures* for a consensus set. A first reliability measure for a transformation is the consensus measure (section 6.2.1). In itself, this measure is not sufficient because there can be multiple candidate matches for each feature (including false matches), which is especially a problem for large RoIs. Therefore, we consider other reliability measures in this chapter.

We expect to find a consistent set of transformation parameters for all features that move consistently on the same object (as in Figure 8.1). Therefore, by demanding *parametric consistency*, we can obtain a reliable *consistent consensus set*. We will introduce an *intersection graph* representation for the mutual consistency of polytopes, from which we can derive whether there is a common intersection of many TUPs.

Additionally, we examine our *confidence* in a consensus set for the given data distribution. Assume a situation where features are randomly distributed over the image and a sample set is randomly chosen, then what is the probability that a consensus set of a certain size arises? That is, what is the probability that a consensus set of that size can occur by accident in random data? We will show how to derive this probability of an *accidental consensus set* of any size, given a random data distribution. For a real (non-random) data distribution we expect that the distribution of part of the data (the inliers) in a correspondence problem is caused by some transformation. Then, the occurrence of a (large) consensus set obtained for an all-inlier sample set cannot be explained by a random distribution of data in the image. Therefore, if we find a consensus set that is highly unlikely to occur by accident, it must be explained well by some transformation, and we are confident that it is a reliable solution.

In most robust model fitting methods, hard thresholds (determined by a priori estimations about the data) are applied to determine the confidence level. Our approach is related to the so-called *a contrario* reasoning or *Computational Gestalt Theory* (CGT) [Sur et al., 2008]. Generally, these methods rely on the assumption that a pattern in a data set is significant if its density is so high that such an arrangement is unlikely to be due to randomness. CGT has already proved its efficiency in problems such as motion detection [Veit et al., 2006], segmentation [Burrus et al., 2009], or shape matching [Musé et al., 2006].

Instead of using necessarily incomplete and approximative models, a contrario methods rely on a generally trivial model of pure chance, called the *a contrario or noise model*. Then the events of interest are those

whose probability to occur in pure noise is very low, according to the so-called Helmholtz principle. In other words, there must be a better explanation for the observed pattern than randomness. The main interest of this approach comes from its ability to accurately compare event significance by simple a contrario models, which allows to automatically determine thresholds, and thus avoids parameter tuning.

The following section considers the verification of consistency in the transformation parameter domain. Section 8.3 introduces a confidence measure that evaluates the reliability of the consistent correspondence set. The introduction of the reliability measures into our matching framework, both as heuristic and as confidence measure, is illustrated by experimental results on image sequences in different applications. The consistency constraints and confidence measures are successfully involved in the solution of correspondence problems.

8.2 Consistency

Figure 8.2 gives a sketch of a correspondence problem where feature displacement is small in between consecutive frames, e.g., as one can expect for the tracking application of Figure 8.1. Both images are represented by a set of features, $\mathbf{x_i} \in S$ and $\mathbf{x'_{ij}} \in S'$ (here in the same axes). We assume that the candidate matches in S' must occur in a local neighborhood of the features in S. Typically, for each point $\mathbf{x_i}$, there are multiple candidate correspondences, i.e., a set Q'_i of m putative correspondences $\mathbf{x'_{i1}}, \ldots, \mathbf{x'_{im}}$ located in a search region centered on the $\mathbf{x_i}$. We must determine the correct correspondence for each $\mathbf{x_i}$ from Q'_i.

To solve such a correspondence problem, we will present a matching procedure that involves finding TUPs that have both a high consensus measure as well as a reduced level of uncertainty. Therefore, we first introduce 2 types of uncertainty regions. A first type covers the region in which the candidate matches must be found, a second type represents the localization uncertainty introduced during feature detection.

First, to compute a *global constraint region* R'_i (GCR) in the second image where the candidate matches for $\mathbf{x_i}$ must be located, we include prior known bounds on the transformation parameters. For example, in tracking tasks we know that the match in the next image must be found close to the current location of the feature. Or, when the change in position, orientation and focal distance of a camera in between the

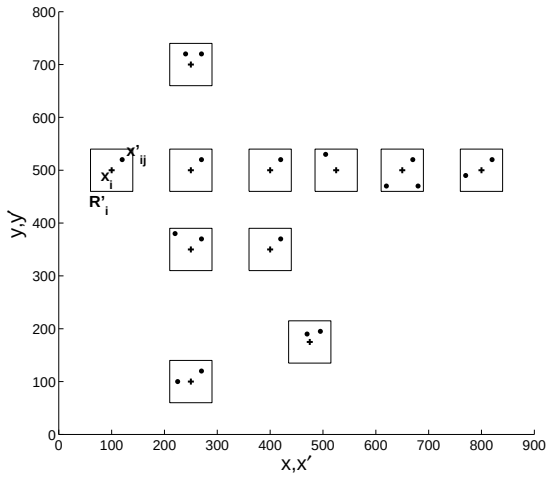


Figure 8.2: A sketch of a correspondence problem as illustrated in Figure 8.1. Determine the set of inliers for the actual transformation of a feature set S (crosses) to the candidate matches in S' (dots). Each GCR encloses a set Q'_i of candidate matches $\mathbf{x'_{ij}} \in S'$ for a feature $\mathbf{x_i} \in S$. The localization uncertainty for each candidate match is modeled by CURs.

acquisition of 2 images are more or less known in advance (e.g. for a stereo pair), then we can constrain the TUP in the transformation space. In those cases, we can define a *global constraint polytope* $\mathcal{T}_g$ (GTUP).

We consider the feature detector to be not completely accurate, that is, even after adequately transforming $\mathbf{x_i}$ to $\hat{\mathbf{x}}'_\mathbf{i} = \mathsf{T}\mathbf{x_i}$, the corresponding feature point in another image may still be displaced a few pixels from a candidate match $\mathbf{x'_{ij}}$. This is modeled by a small polygonal *correspondence uncertainty region* R'_{ij} (CUR) around each $\mathbf{x'_{ij}}$ (with $R'_{ij} \subset R'_i$). We can compute a more accurate estimation of the transformation by adding the inequalities for each correspondence pair $(\mathbf{x_i}, R'_{ij})$, and the GTUP is further reduced to a *correspondence polytope* $\mathcal{T}_{ij}$ (CTUP).

To solve the correspondence problem we must verify whether there is a subset of correspondences that are spatially consistent. Our matching procedure verifies consistency in the parameter space. Two CTUPs are *mutually consistent* when both polytopes intersect, i.e., there is a common set of transformations in parameter space. The remainder of this section presents each building block of our matching procedure in more detail.

8.2.1 Constraint Polytopes

The prior constraints on the transformation parameters define a first TUP. Suppose we know that the transformed image of each point $\mathbf{x_i}$ is confined to a convex polygonal GCR R'_i. Then the transformation must be contained in the GTUP $\mathcal{T}_g = \cap_i \mathcal{T}(\mathbf{x_i}, R'_i)$. The requirement that the prior constraints on the transformation are formulated in terms of GCRs R'_i is not so important. A GTUP $\mathcal{T}_g$ could have been given instead, for which the IURs $R'(\mathbf{x_i}, \mathcal{T}_g)$ are computed if necessary.

In many applications, we can naturally deduce the GTUP. Either we deduce the constraints from a simple *test procedure*, or from a (simplified) *model* of the scene. In the first case, suppose that we expect much vibration or turbulence on an otherwise static camera. We could acquire a short sequence in representative circumstances, and estimate the GCRs. For example, if we place a checkerboard in front of the camera, then the position of the corner points can be easily indicated in consecutive frames. An example is shown in Figure 8.3(a). All detected positions are superimposed on the first image, and polygonal GCRs can easily be deduced.

In the second approach, a pinhole camera model (section 4.4) is constructed to represent the acquisition process (Figure 8.3 (b)). The camera vibration can be represented in terms of bounds on the translation and orientation parameters in a perspective transformation (4.10). We consider the vibration as a convex set of transformations representing small 3D translation and rotation (here mostly around the optical axis) of the camera. If we assume that the interesting features in the scene are located on a world plane for which the orientation and position are known (e.g. $Z_w = 0$ in Figure 8.3 (b)), we can project the features onto the camera image by all perspective transformations in the set. This results in the polygonal regions in Figure 8.3 (c). These regions define the bounds on the GTUP for this application.

As a second example, suppose we know the acquisition set-up, e.g., for Figure 8.4 where a static camera observes a parking lot. The application must keep track of the vehicles driving up and down the lane, as shown in the example frames (a). If the camera parameters are known (albeit up to some uncertainty), then we can construct a model environment as in (b), and eventually define a GTUP. The maximal displacement of a point on a vehicle in between 2 frames is dependent on the expected speed of the vehicles at that world location, and can be modeled as a convex region in a world plane (c). Then we can use the

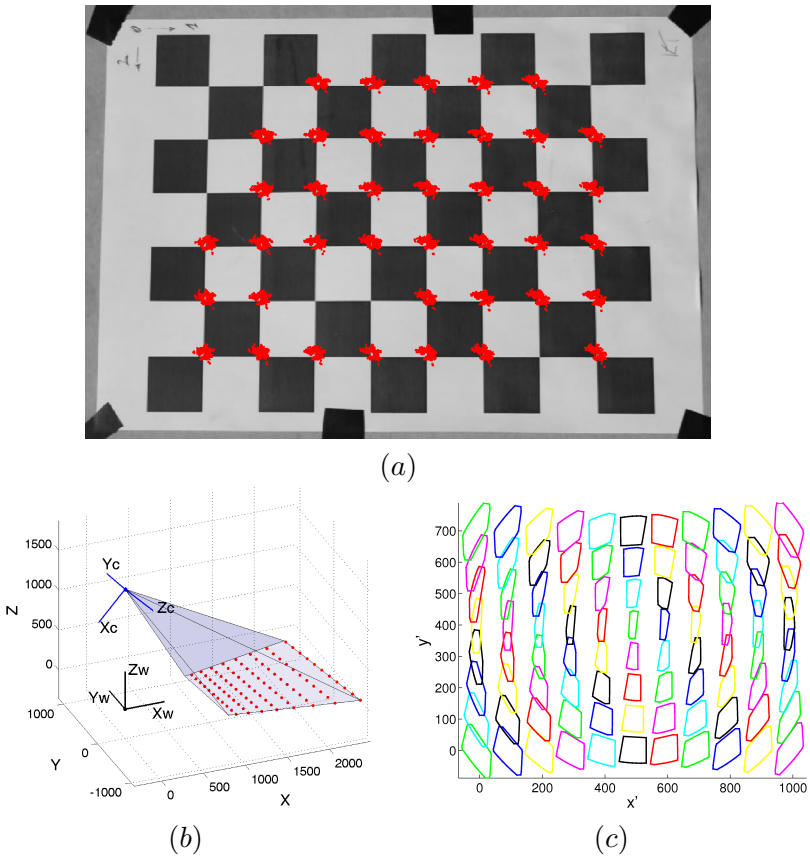


Figure 8.3: The estimation of a GTUP, by in-situ measurement (a), or by modeling $(b-c)$.

perspective projection (4.10) to map this shape onto the image plane, as a representation for the GCRs (d).

In the case of dynamic cameras, e.g., when mounted on a moving vehicle, we can generate a simplified model of the imaged environment as in Figure 8.5 (a). A typical scene model consists of 4 planes (b): a ground plane, two parallel planes (to represent objects at each side of the road) and a frontal plane (for objects in front of the camera). The visual flow pattern defines the locations for the features in consecutive images assuming a constant linear motion pattern for the camera. If we superimpose an estimated vibration pattern on these locations, we can derive the GCRs (c), and the GTUP. In our opinion, it is possible to derive sufficiently accurate GCRs for most applications.

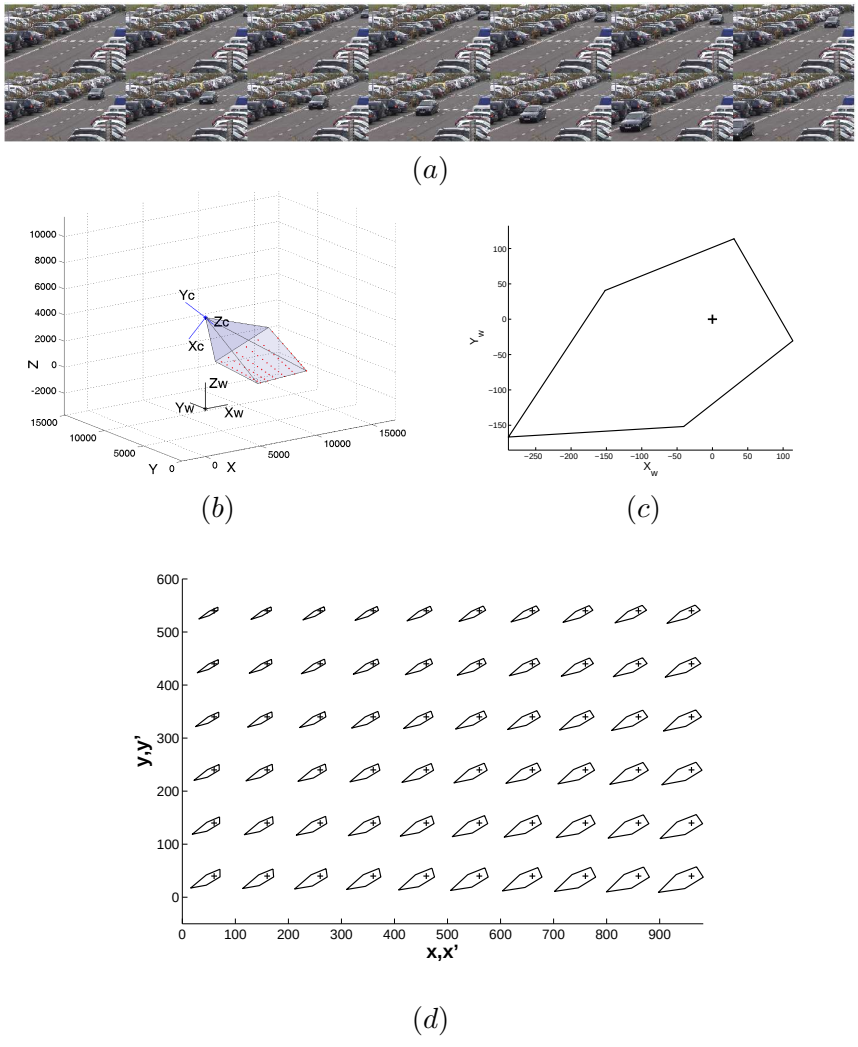
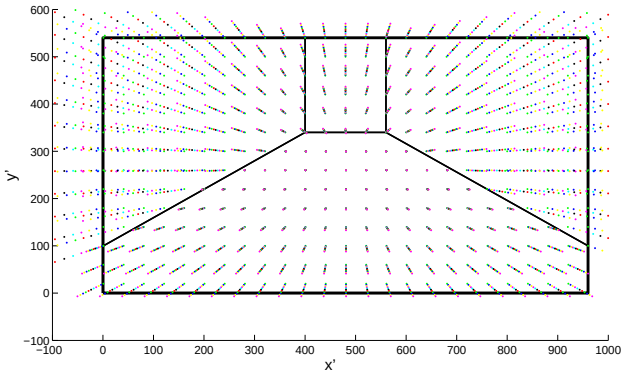
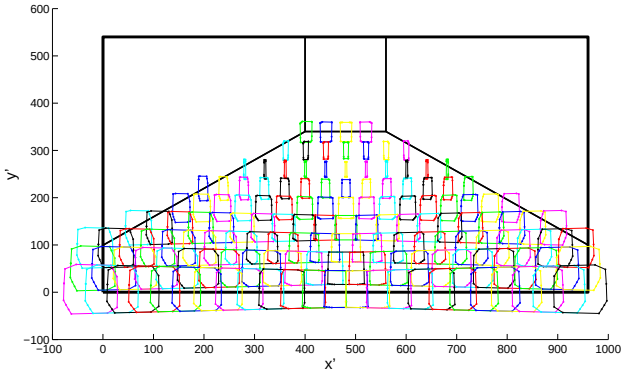


Figure 8.4: GTUP estimation for the tracking of vehicles observed by a static camera (a). (b) Model the camera and the 3D environment. (c) Represent possible locations of candidate matches in frame $(t+1)$ for a feature in frame (t) (+) as a polygonal region in a world plane $Z_w = 0$ (e.g. in terms of driving direction and speed). (d) Map these regions onto the image for a sampled set of features.

(a)



(b)



(c)

Figure 8.5: Estimating the GTUP for dynamic cameras, e.g., mounted on a moving vehicle (a). (b) The location of features is expected to change as illustrated by the visual flow for a constant linear motion of the camera. (c) An illustration of the GCRs when the camera is vibrating.

8.2.2 Correspondence Polytopes

In each GCR R'_i we find (multiple) candidate matches $\mathbf{x'_{ij}}$ for a feature $\mathbf{x_i}$. We define the CUR R'_{ij} to take the inaccuracy of the feature localization into account. Section 2.4 has shown that square regions of few pixels high suffice to represent most of the small displacement errors for each candidate image point $\mathbf{x'_{ij}}$.

Suppose we use a square GCR R'_i of height τ centered on each point $\mathbf{x_i}$ to confine the possible locations of candidate matches $\mathbf{x'_{ij}}$. As shown in Figure 8.6 (a), R'_i contains all possible candidate matches $\mathbf{x'_{ij}}$ in the second image. The correspondence region is a small square R'_{ij} of height ϵ, with $\epsilon < \tau$, and the CTUP $\mathcal{T}_{ij} = \mathcal{T}_g \cap \mathcal{T}(\mathbf{x_i}, R'_{ij})$.

Then the IURs must satisfy $R'(\mathbf{x_i}, \mathcal{T}_g) \subseteq R'_i$, as well as $R'(\mathbf{x_i}, \mathcal{T}_{ij}) \subseteq R'_i$. Figure 8.6 (a) shows the IURs $R'(\mathbf{x_i}, \mathcal{T}_{ij})$ of one CTUP $\mathcal{T}_{ij}$.

8.2.3 Mutual Intersection of Correspondence Polytopes

If two CTUPs $\mathcal{T}_{ij} = \mathcal{T}_g \cap \mathcal{T}(\mathbf{x_i}, R'_{ij})$ and $\mathcal{T}_{kl} = \mathcal{T}_g \cap \mathcal{T}(\mathbf{x_k}, R'_{kl})$ are given, $\mathcal{T}_{ij}$ and $\mathcal{T}_{kl}$ are denoted as *mutually consistent* provided the *polytopes intersect*, i.e., $\mathcal{T}_{ij} \cap \mathcal{T}_{kl} \neq \emptyset$.

If both polytopes intersect, let $R'_{ij,kl}(\mathbf{x_i}, \mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ denote the region implied by $\mathcal{T}_{ij} \cap \mathcal{T}_{kl}$ for $\mathbf{x_i}$. The CTUPs $\mathcal{T}_{ij}, \mathcal{T}_{kl}$ in Figure 8.6 (b) are computed for both indicated point-region pairs. All other points $\mathbf{x_m}$ are mapped into IURs $R'_{ij,kl}(\mathbf{x_m}, \mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ as shown in Figure 8.6 (b). Then there is a consensus set $C(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ that can be explained by at least one transformation in the non-empty intersection.

Note that IURs for the intersection of TUPs are always smaller than those for an individual TUP. By definition, $R'_{ij,kl}(\mathbf{x_m}, \mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ must be a subset of both $R'(\mathbf{x_m}, \mathcal{T}_{ij})$, and $R'(\mathbf{x_m}, \mathcal{T}_{kl})$, i.e., the IURs for 2 intersecting CTUPs will always be contained in those of a single CTUP [Veelaert and Teelen, 2006a]. Thus, the uncertainty is reduced by combining the information.

8.2.4 Intersection of N correspondence polytopes

We will now derive more precise results concerning the consensus set for the intersection of more than 2 CTUPs. When we find a non-empty intersection of N CTUPs $\mathcal{T}_{ij} \cap \ldots \cap \mathcal{T}_{nm} \neq \emptyset$, we have proven that there must be a TUP with a consensus measure of at least N [Veelaert and

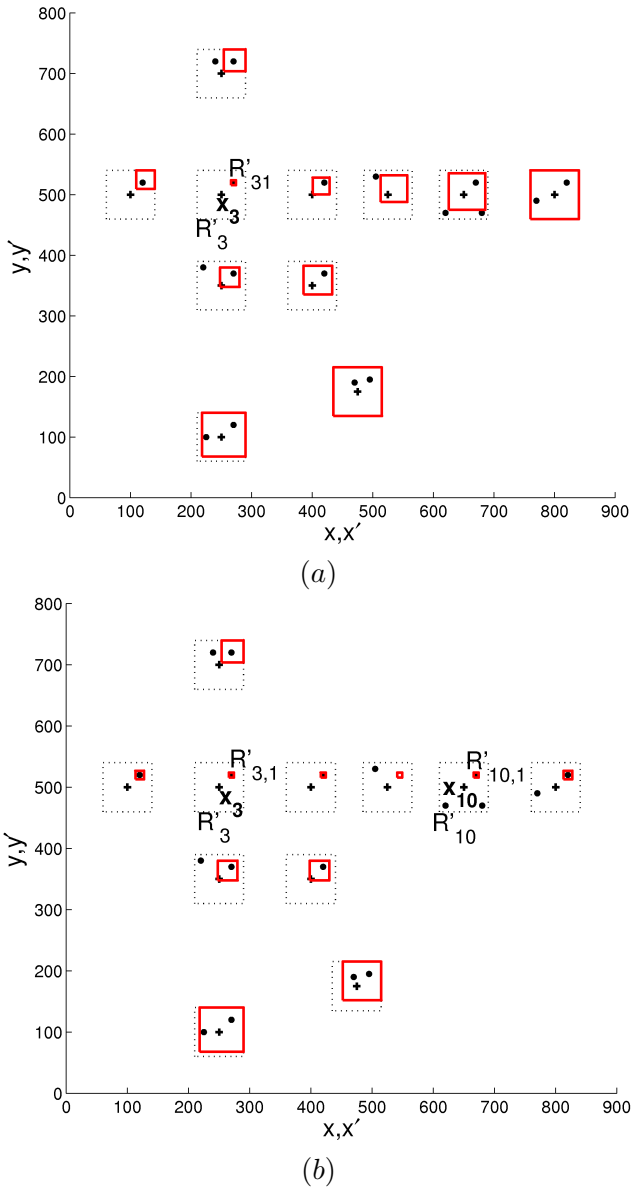


Figure 8.6: The CTUP $\mathcal{T}_{3,1}$ is defined by the inequalities from the GCRs R'_i for the points $\mathbf{x_i}$ and a smaller CUR $R'_{3,1} \subset R'_3$ centered on a candidate match $\mathbf{x'_{3,1}}$. (a) The IURs $R'(\mathbf{x_i}, \mathcal{T}_{3,1})$ (solid lines) are always subsets of the GCRs R_i (dashed lines). (b) The intersection of the CTUPs $\mathcal{T}_{3,1} \cap \mathcal{T}_{10,1}$ is non-empty and is used to compute the IURs $R'(\mathbf{x_i}, \mathcal{T}_{3,1} \cap \mathcal{T}_{10,1})$ (solid lines). Notice that the area of the IURs in (b) is smaller than that of the IURs in (a).

Teelen, 2006a]. Hence, we should be looking for non-empty intersections of CTUPs, and the most straightforward tool to do so is an intersection graph.

The Intersection Graph

The intersection graph is constructed as follows.

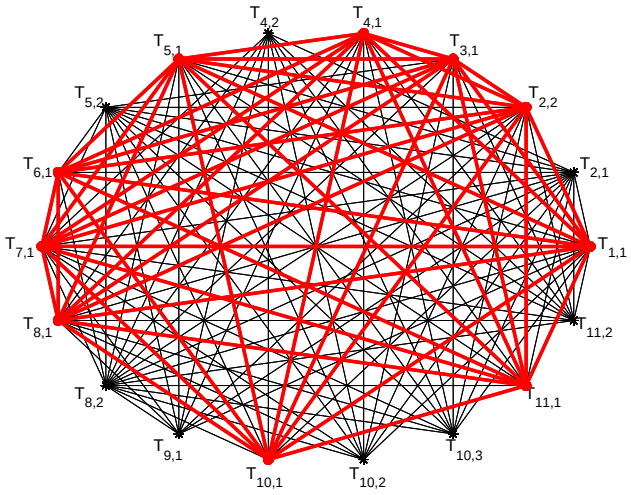
Definition 8.1 *Suppose the points* $\mathbf{x'_{ij}}$*, the CURs* R'_{ij}*, and the CTUPs* $\mathcal{T}_{ij} = \mathcal{T}_g \cap \mathcal{T}(\mathbf{x_i}, R'_{ij})$ *are defined as above. The intersection graph* $G(\mathcal{T}_{ij})$ *represents each polytope* $\mathcal{T}_{ij}$ *by a vertex. Two vertices are joined by an edge when the intersection of both CTUPs is non-empty.*

The intersection graph $G(\mathcal{T}_{ij})$ for the situation in Figure 8.6 is shown in Figure 8.7 (a). Notice that the CTUPs $\mathcal{T}_{3,1}$ and $\mathcal{T}_{10,1}$ from Figure 8.6 are connected by an edge in $G(\mathcal{T}_{ij})$. The presence of an edge in $G(\mathcal{T}_{ij})$ corresponds to a non-empty intersection of *three* polytopes (not two), that is, $\mathcal{T}_g \cap \mathcal{T}(\mathbf{x}_i, R_{ij}) \cap \mathcal{T}(\mathbf{x}_k, R_{kl}) \neq \emptyset$.

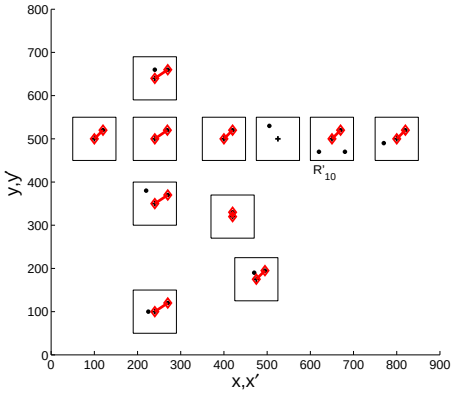
Furthermore, the intersection graph is also known to be a r-partite graph, with r classes of vertices $\{\mathcal{T}_{k1}, \mathcal{T}_{k2}, \ldots, \mathcal{T}_{kn}\}, 1 \leq k \leq r$. r is the number of features $\mathbf{x_i} \in S$, and n the number of candidate matches for $\mathbf{x_i}$. For example, consider the situation in Figure 8.6. There are $r = 11$ features in S, and $n = 3$ candidate matches in R'_{10}, which is reflected in the intersection graph of Figure 8.7 (a).

Now, we want to detect the maximum number of CTUPs that mutually intersect each other. A set of intersecting CTUPs can be found as a *clique* in the intersection graph $G(\mathcal{T}_{ij})$. A clique is a fully connected or complete subgraph in which each pair of graph vertices is connected by an edge. A clique is *maximal* if it is not part of a larger clique in the intersection graph, and the *maximum* clique is the largest maximal clique in the graph. Figure 8.7 (a) shows the maximum clique for $G(\mathcal{T}_{ij})$ by the thicker lines.

In a correspondence problem, we expect to find one transformation for a large(r) fraction of correspondence pairs, thus their CTUPs are expected to intersect. So the intersection graph $G(\mathcal{T}_{ij})$ usually contains a single large clique, to which some additional vertices are loosely connected. That is, there is a subset of vertices that have large degree (the number of edges at a vertex) forming a clique, and additional vertices that have low degree. The loosely connected vertices with low degree are explained by the CTUPs for candidate pairs that intersect with some



(a)



(b)

Figure 8.7: (a) The intersection graph $G(\mathcal{T}_{ij})$ for the situation in Figure 8.6 with an overlay (thicker edges) of the detected maximum clique. (b) The correspondence pairs in the CCS, which are determined by finding the maximum clique, are connected by solid lines.

but not all polytopes in the clique (depending on the location of the features).

From now on, the non-empty intersection of all CTUPs in the maximum clique is denoted as the *consistent intersection polytope* (CITUP).

The correspondence pairs $(\mathbf{x_i}, \mathbf{x'_i})$ in the consensus set of the CITUP will be referred to as the *consistent consensus set* (CCS). The features in the CCS are connected in Figure 8.7 (b). If we expect to find multiple CCSs in an application, then there should also be multiple larger maximal cliques in the intersection graph.

Computing Cliques

Many combinatorial optimization problems for general graphs are NP-complete, and so are maximum clique algorithms. With standard algorithms, large computational effort is required to find a maximum clique in a large random graph (> 50 vertices) [Valiente, 2002]. However, the graphs that appear in uncertain geometry are far from being random graphs [Veelaert, 2002], and so is the intersection graph, as explained above. This has important consequences from a computational viewpoint, since the graph properties can be used to design a more efficient clique finding algorithm.

First, because it is a r-partite graph, a considerable speed-up can be achieved for a maximum clique algorithm, since it must only look at subgraphs that contain at most 1 vertex of each subset in the partition. Second, we can compute an upper and lower bound, b_u and b_l, on the maximum clique size in advance. This again reduces complexity because we must not consider complete subgraphs of a size smaller than b_l, and must not extend complete subgraphs to a size larger than b_u.

Computing the lower bound. *Turan's Theorem* is used to compute a *lower* bound b_l on the maximum clique size. According to Turan's Theorem a graph with $|V|$ vertices *without* a clique of size $p, p > 1$, can have at most

$$\left(1 - \frac{1}{p-1}\right) \frac{|V|^2}{2} \tag{8.1}$$

edges. Given $|V|$ and the number of edges $|E|$ in G, we can obtain the lower bound b_l on the clique size. Furthermore, we can improve this lower bound by eliminating one by one the vertices of minimal degree from the graph G and recalculating the minimal clique size for each subgraph until (8.1) exceeds the number of edges left.

Computing the upper bound. To derive an *upper* bound b_u for the maximum clique size, first choose the vertex with maximal degree from

each class of the r-partite graph G. Arrange these r vertices according to decreasing degree, and denote the ordered sequence by $d_1, d_2, \ldots, d_r$. Then the upper bound is given by the maximum value of the index k such that $d_k \geq k-1$, since a clique of size k requires the presence of at least k vertices of degree $d_k \geq k-1$.

Then, if $b_u > b_l$, consider all subgraphs of order b_l of G, and verify for each subgraph whether it is a clique. We try to extend each of these cliques by adding one vertex at a time. Note that the difference $b_u - b_l$ is a good indication of the computational effort that will be needed to find a maximum clique.

8.2.5 Analyzing the Intersection Graph

From the intersection graph we want to derive whether there is a set of N intersecting CTUPs. Since any pair of vertices in a clique is joined by an edge, it follows that $\mathcal{T}_{kl} \cap \mathcal{T}_{mn} \neq \emptyset$ for any pair of polytopes of a clique. However, this does not imply that $\mathcal{T}_{ij} \cap \ldots \cap \mathcal{T}_{uv} \neq \emptyset$ for a maximal clique. In some cases, the polytope $\mathcal{T}_{ij} \cap \ldots \cap \mathcal{T}_{uv}$ can be non-empty as shown by the following theorem.

Theorem 8.2 (Helly) *Let* $\mathbf{F}$ *be a family of at least* $d+1$ *polytopes in* $\mathbb{R}^d$*. If every subset of* $d+1$ *polytopes in* $\mathbf{F}$ *has a non-empty intersection, then the intersection of all polytopes in* $\mathbf{F}$ *is non-empty.*

Theorem 8.2 is a version of Helly's famous theorem for convex sets, here restricted to polytopes [Helly, 1923; Rockafellar, 1970; Stoer and Witzgall, 1970]. Only for one-dimensional polytopes (intervals) a clique in an intersection graph corresponds to a common non-empty intersection [Veelaert, 1999b]. For d-dimensional sets ($d > 1$) a clique in the intersection graph is not sufficient as it considers only the intersection of every pair of polytopes.

Not only the correctness of the result but also the computational effort matters. Recall that the intersection graph already represents not just the intersection of 2 polytopes, but 3, as all CTUPs must be a subset of the GTUP. Even though an intersection graph may give incomplete information about the common intersections of N d-dimensional polytopes ($d > 1$), it may still be useful as an indication, as long as it is possible to verify afterward whether the common intersection is non-empty. If the maximum clique in an intersection graph $G(\mathcal{T}_{ij})$ indeed

corresponds to an empty intersection, then one can explore subgraphs of the maximum clique, or the second largest maximal clique.

We can also construct alternative types of intersection graphs that allow us to find a clique that *does* correspond to a non-empty intersection of polytopes, even in $\mathbb{R}^d$, $d > 1$. We will construct an intersection graph $G(\mathcal{T}_{ij} \cap \ldots \cap \mathcal{T}_{mn})$ where each vertex does not represent 1 polytope, but the intersection of multiple polytopes. Then each edge represents whether the intersection of multiple intersection polytopes is non-empty.

For instance, when considering transformations of the form

$$\begin{cases} x' & = a_1 x + a_3 \\ y' & = a_5 y + a_6 \end{cases}, \tag{8.2}$$

we can represent the polytopes in 2 2D parameter spaces. According to Helly's Theorem a non-empty intersection of 2D polytopes corresponds to a clique in a 3-uniform intersection hypergraph. In a 3-uniform hypergraph each hyperedge is of size 3, i.e., each hyperedge represents a non-empty intersection of 3 polytopes. Let H be a 3-uniform intersection hypergraph, with n vertices v_i, one for each $\mathcal{T}_i$. Each hyperedge indicates a non-empty subset of vertices $\{v_i, v_j, v_k\}$ such that $\mathcal{T}_i \cap \mathcal{T}_j \cap \mathcal{T}_k \neq \emptyset$.

Figure 8.8 (b) gives an illustration of a 3-uniform intersection hypergraph H with vertices for each polygon in the set $S = \{A, B, C, D, E\}$ (a). The hyperedges in H represent whether a subset of 3 polygons intersects in (a), and are indicated by closed curves in (b). Thus, the set of hyperedges is $\{\{A, B, C\}$,$\{A, B, D\}$,$\{A, C, D\}$,$\{B, C, D\}$,$\{B, C, E\}\}$.

We can now construct an intersection graph $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ where each vertex denotes the intersection of 2 TUPs. Then $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ has $n(n-1)/2$ vertices denoted as $v_i v_j$. The edges of $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ consist of all vertex pairs $\{v_i v_j, v_k v_l\}$ satisfying $\mathcal{T}_i \cap \mathcal{T}_j \cap \mathcal{T}_k \cap \mathcal{T}_l \neq \emptyset$.

Figure 8.8 (c) illustrates how the graph H with 5 vertices is replaced by a graph G with 10 vertices $\{AB, AC, \ldots\}$. The graph G actually shows for each intersection of 2 polygons in S whether it intersects with another intersection of 2 polygons in S.

The motivation to replace the hypergraph H by the graph $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ is as follows. Suppose we find a clique in $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ with m vertices $v_i v_j, \ldots, v_m v_n$. For each edge $\{v_i v_j, v_k v_l\}$, $\mathcal{T}_i \cap \mathcal{T}_j \cap \mathcal{T}_k \cap \mathcal{T}_l \neq \emptyset$. Therefore $\mathcal{T}_i \cap \mathcal{T}_j \cap \mathcal{T}_k \neq \emptyset$, …, $\mathcal{T}_j \cap \mathcal{T}_k \cap \mathcal{T}_l \neq \emptyset$ for the 4 combinations of 3 sets in a collection of 4.

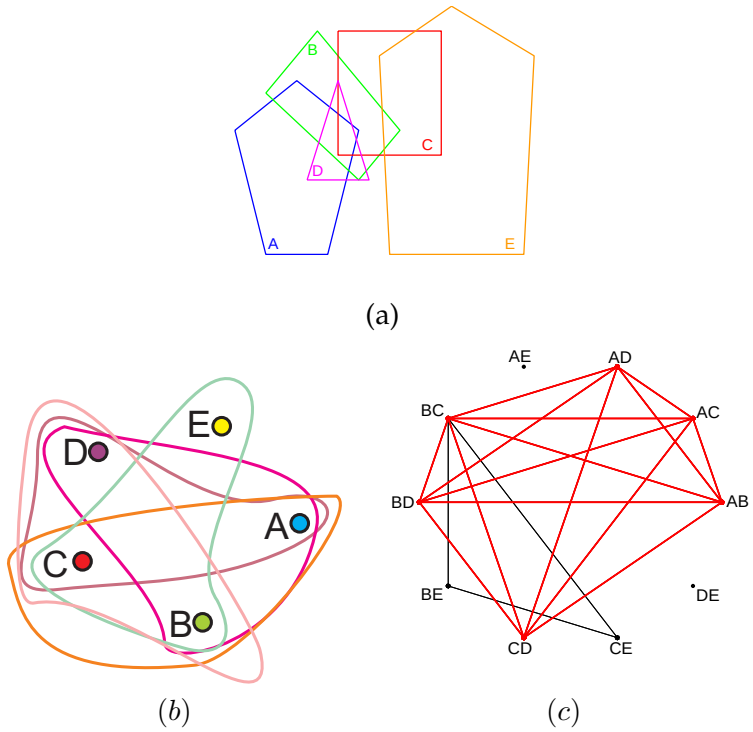


Figure 8.8: (a) A set of polygons. (b) The hyperedges in the 3-uniform intersection hypergraph H represent the common intersection of 3 polygons in (a). (c) The hypergraph H is replaced by an intersection graph G where each vertex represents the common intersection of 2 polygons.

Since the clique has $m(m-1)/2$ edges this implies that for the collection $C = \{\mathcal{T}_i, \mathcal{T}_j, \ldots, \mathcal{T}_m\}$ there are at least $4m(m-1)/2$ intersections among the $2m(2m-1)(2m-2)/6$ possible intersections of 3 arbitrary sets in C that are known to have a non-empty intersection. That is, the ratio of triples that have been verified to the total amount of triples is $6(m-1)/((2m-1)(2m-2))$. This means that if one finds a clique in G for which m is not too large, there is a good chance that it also corresponds to a clique in the hypergraph H.

In the graph G of Figure 8.8 (c), we find a maximum clique of 6 vertices. The maximal clique in H consists of 4 vertices $\{A, B, C, D\}$. Even though clique finding in hypergraphs is a complex problem, we can conclude that our approach reduces the overall computational complexity considerably, at the cost of occasional empty intersections.

8.2.6 Evaluation of the Matching Procedure

The maximum clique in an intersection graph $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$ assists in discriminating the CCS in a correspondence problem. However, we do not only want to know the probability that the intersection of N TUPs is empty. We also want to verify whether the returned CCS is correct. An analysis of the results in a controlled set-up during a Monte Carlo simulation allows us to verify whether and to what extent different parameters influence the performance of our matching framework.

In each set of experiments, the following parameters are varied:

- the number N of feature points $\mathbf{x}_i \in S$ in the first image;
- the height τ of the square GCRs R'_i, which determine the size of the GTUP $\mathcal{T}_g$;
- the height ϵ of the square CURs R'_{ij}, which determine the size of the CTUP $\mathcal{T}_{ij}$;
- the average fraction of false positives FP and false negatives FN among the candidate matches $\mathbf{x}'_{ij} \in S'$ in the second image. FP and FN are given as a percentage of N.

For each set of $n = 80$ experiments, we choose a fixed set of parameters $[N, \tau, \epsilon, FP, FN]$. In each of the experiments, we randomly generate a set S of feature locations in the first image (of size 800×600). We establish a ground truth correspondence set by transforming on average $(1 - FN/100)N$ points to their actual matches within the GCRs R'_i. The location of these matches is randomly varied around the actual location within R'_{ij}. We generate on average $(FP/100)N$ false candidate matches in the regions $R'_i \backslash R'_{ij}$. Figure 8.9 illustrates such a set-up for the parameter set $[N, \tau, \epsilon, FP, FN] = [14, 40, 3, 200, 20]$.

The parameters for FP and FN determine the number of features in the GCRs, summarized by the *feature density* FD. FD relates the number of features to the total number of points in an image, and can easily be related to the outcome of a feature detector in real applications. FD is expressed as the percentage of all image pixels that is a feature, e.g., 400 features in an image of size 800×600 gives $FD = 0.08\%$.

All experiments are evaluated by the resulting CCS and the non-emptiness of the CITUP $\mathcal{T}_C = \bigcap \mathcal{T}_{ij}$. $\mathcal{T}_C$ is computed as the intersection of all CTUPs $\mathcal{T}_{ij}$ in the maximum clique of the intersection graph, as explained in section 8.2.5. For each set of experiments is indicated in how many percent of the experiments a non-empty $\mathcal{T}_C$ occurs. In each experiment we compute the CCS $C(\mathcal{T}_C)$, and verify whether it is a sub-

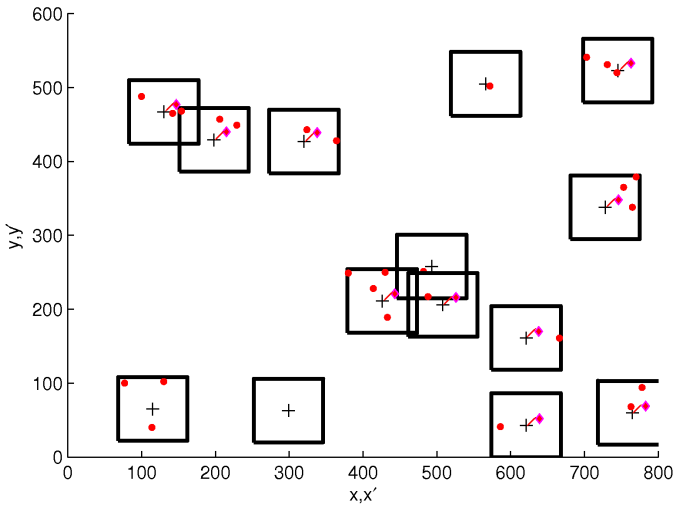


Figure 8.9: Monte Carlo simulation set-up.

set of the ground truth correspondence set. We aim to answer the following questions. How many of the ground truth correspondences are correctly determined? And secondly, how many of the correspondence pairs in $C(\mathcal{T}_C)$ are not part of the ground truth?

To summarize the evaluation criteria we give a representative subset of results in Table 8.1, illustrating the most important trends, with

- **GT**: the average percentage of correspondence pairs in the CCS that occur in the ground truth set;
- **NGT**: the average percentage of correspondence pairs in the CCS that are *not* in the ground truth set;
- **NEP**: the percentage of experiments with a non-empty CITUP.

The indicated results are averaged over all experiments conducted with the same parameter set. Figure 8.10 presents the described experiments in the table visually, and for a wider variation of parameters (indicated in the graph legends).

From the presented results, we can conclude that our method succeeds in extracting the ground truth correspondence set for most parameter combinations, without making much mistakes.

The CITUP is non-empty in most cases. In a few experiment sets, there are some (up to 5%) cases where the CITUP is empty. This occurs

Table 8.1: An evaluation of the Monte Carlo simulations for intersection graphs.

	N	τ	ϵ	FP	FN	FD	**GT**	**NGT**	**NEP**
A	12	50	5	50	30	0.05	95.08	2.69	100
				100		0.07	95.91	0.95	100
				150		0.09	94.87	0.25	100
				200		0.11	90.34	4.10	95
				250		0.13	88.20	5.82	95
				300		0.15	84.59	8.35	90
				350		0.17	78.57	9.1	97.5
				400		0.19	80.72	10.2	97.5
B	12	45	5	50	30	0.06	92.94	1.99	100
				100		0.09	95.90	0.61	100
				150		0.11	91.40	0.95	100
				200		0.14	85.68	6.59	97.5
				250		0.17	83.71	8.33	97.5
				300		0.19	84.46	5.31	97.5
				350		0.22	81.11	10.11	95
				400		0.24	80.67	7.87	95
C	12	45	3	200	20	0.14	90.33	0	100
					30	0.14	87.99	0.77	100
					40	0.13	87.18	4.75	100
					50	0.13	87.31	6.13	97.5
					60	0.12	84.22	11.20	100
D	16	45	3	50	30	0.06	95.14	0.96	100
				100		0.09	93.54	0.90	100
				150		0.11	92.43	1.92	97.5
				200		0.14	87.53	3.12	97.5
				250		0.17	89.62	2.50	100
				300		0.19	84.38	4.33	97.5
				350		0.22	77.63	10.01	95
				400		0.24	81.87	4.55	97.5

mostly when there is a high percentage of FP, and/or for a higher number N of feature points. As long as the probability of detecting an empty CITUP is small, the probability of false information is low. The worst thing that can happen in a real application (as long as it is easy to verify the non-emptiness of the CITUP), is that one must try the second largest maximal clique, or start over with a new set of features

and candidate matches.

In experiments **A** and **B** of Table 8.1, the number of FP is increased, and thus also the FD. We notice that the CCS is correctly obtained in almost any case, but on average only $85 - 95\%$ of all correspondences in the GT set are obtained. In most experiments on average just a few percent of false matches are returned in the CCS.

We notice that FD exerts an important effect on the outcome of the experiments. The higher FD, the less reliable the results of our method. If there is a substantial number of features in the GCR (for higher rates of FP), our matching method can deviate slightly from the ground truth solution. However, as these are artificial experiments, we do not worry too much about this result, as in real images we do not expect false candidates to be located as close to the actual correspondence as in these artificial experiments. Typically, the outcome of the feature detector can be related to the FD.

The graphs in Figure 8.10 $(a - d)$ show the same trend for experiments with other parameters. From Figure 8.10 $(c - d)$, we can deduce that the size of the CURs (ϵ) also is an important parameter, according to the experiments **D**. If the uncertainty about the feature localization (thus ϵ) increases over a few pixels, then our procedure will fail more often as the CTUPs become too large. Thus, if the feature detector is not sufficiently reliable, the matching procedure will not be that reliable.

Also the size of the GCRs (τ) is important as it determines the size of the global transformation window. We can assume that the matching process benefits from an accurate computation of the RoIs, because the larger the GCR, the higher the percentage of FP in a real application. For an increasing size of GCR (and other parameters fixed), we observe from Figure 8.10$(a - b)$ that the consensus set is more reliable for the larger GCRs, because FD is lower.

If not enough correct correspondences are available in the data set (i.e. the number of FN is rising), then we will most likely not obtain a correct CCS by computing the clique in the intersection graph. An illustration is given in experiments **C** and Figure 8.10$(e - f)$, where we see that the performance of the matching algorithm deteriorates for increasing number of FN. When the number of FN exceeds 60% then our method fails regularly, because of the lack of correct matches. Therefore, it is vitally important in real applications to be able to select features for which correspondences do occur in other images, i.e. the repeatability rate must be high.

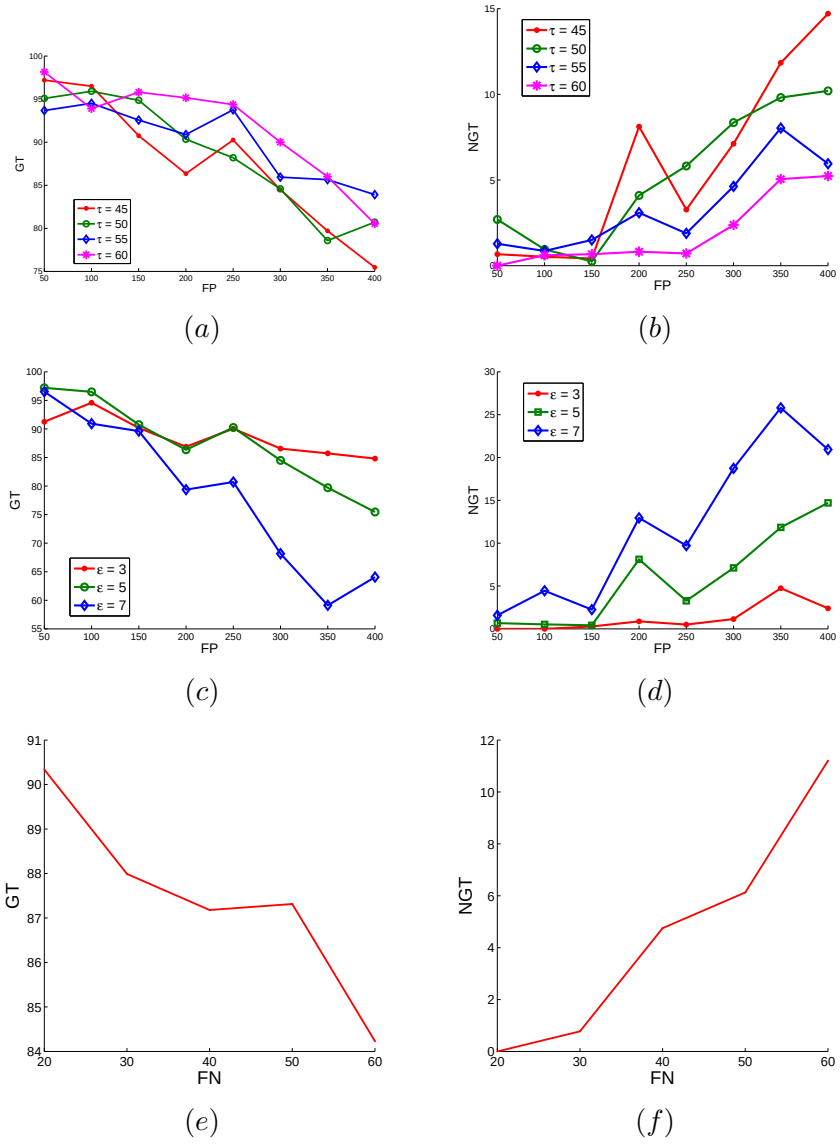
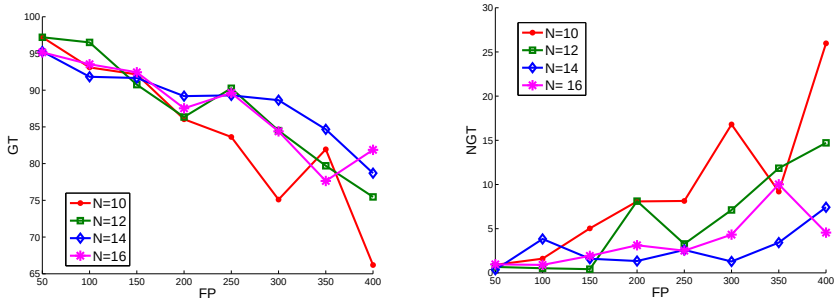


Figure 8.10: Figure continues on next page.



(g) (h)

Figure 8.10: Evaluating the performance of our matching framework:
$(a-b)$ **GT** and **NGT** in function of FP for different values of τ and other parameters fixed as in column **A**;
$(c-d)$ **GT** and **NGT** in function of FP for different values of ϵ and other parameters fixed as in column **B**;
$(e-f)$ **GT** and **NGT** in function of FN and other parameters fixed as in column **C**;
$(g-h)$ **GT** and **NGT** in function of FP for different values of N and other parameters fixed as in column **D**.

From experiment **D**, we can see that the same behavior is expected for different N. There is a (small) improvement in the results for higher N, as expected, but we notice that even for small subsets the results are sufficiently accurate to try them out in real applications.

8.2.7 Conclusion

We must keep in mind that the above evaluation is conducted on artificial data. The results give an adequate impression of the expected results of our matching procedure. However, we expect even better results for real images, because then features must obey the *rules* of the transformations. The actual corresponding feature must be present (except for situations like occlusion), as all features undergo the same transformation (up to some uncertainty about the location). Even more, prior information allows to accurately compute the GCRs, and rule out a considerable number of FP. Thus, we expect that both FP and FN will be rather reduced in real applications. Simulations have shown that under such conditions the probability of detecting a correct CCS is acceptable. We will verify this claim for some examples in section 8.4.

8.3 Confidence

To verify the results obtained by our matching procedure, we propose an additional reliability measure. Whereas the size of the consensus set gives a first indication about its reliability, our *confidence measure* indicates the probability that a consensus set of that size could arise by a random occurrence of features. If the occurrence of such a consensus set *by accident* is unlikely, our confidence in the obtained result increases.

At the same time, we want to determine which are the interesting combinations of two (or more) correspondence pairs as a sample set. We aim at computing a confidence measure that can also be applied as a *heuristic* during the estimation process. This heuristic should reduce the complexity of our search in transformation parameter space considerably, so that better performance can be achieved than for a full blind search for a random consensus.

8.3.1 The Matching Problem Restated

To recapitulate shortly, our matching procedure consists of finding the maximum clique in an intersection graph. When the intersection of all CTUPs in the maximum clique is non-empty, i.e., the $\mathcal{T}_C = T_{ij} \cap \ldots \cap T_{kl} \neq \emptyset$, we have found a CCS $C(\mathcal{T}_C) = \{(\mathbf{x_i}, \mathbf{x'_{ij}}), \ldots, (\mathbf{x_k}, \mathbf{x'_{kl}})\}$, for which the consensus measure $n(C(\mathcal{T}_C))$ is equal to or larger than the size of the clique.

We can now restate our interpretation of the matching problem, and demand for additional confidence. Find a TUP $\mathcal{T}$,

- with a high consensus measure $n(C(\mathcal{T}))$;
- that sufficiently constrains the transformation;
- with a consensus set that did not arise by accident.

The first two requirements are handled by the IURs of $\mathcal{T}$. First, by counting the features contained in the IURs, and second, by looking at their size. The size is also reflected in the intersection graph, as the intersection of multiple TUPs results in IURs of a smaller area. To be precise, we require that the IURs $R(\mathbf{x_i}, \mathcal{T})$ have height at most ϵ (such as the CURs) for features $\mathbf{x_i}$ appearing in the CCS.

To define an additional confidence measure, we consider the probability that an *accidental consensus set* (ACS) of size g (or larger) arises for randomly distributed data. Suppose that the features (both in S and S')

are distributed *randomly* over the entire image according to a uniform distribution. Then how likely is the *accidental* occurrence of a consensus set $ACS(\mathcal{T}_i)$ with consensus measure n_a equal to g or larger, if we compute $\mathcal{T}$ for a randomly chosen sample set of i correspondence pairs $\{(\mathbf{x_i}, \mathbf{x'_{ij}})\}$? That is, compute the probability $P(n_a \geq g)$ that $ACS(\mathcal{T})$ has a consensus measure n_a of size g or larger by coincidence.

The probability $P(n_a \geq g)$ depends strongly on the distribution of the data, thus on

- the density of the feature points in the images, which may include many false positives and false negatives;
- the size of the GCRs (τ);
- the positional inaccuracy of a feature detector, reflected by the size of the CURs (ϵ).

However, in a real matching problem, *not* all data is randomly distributed. A considerable fraction of data points is expected to be explained well by some transformation. If we select a sample set of correspondence pairs related by that transformation from the given data, we expect a larger consensus measure than can be explained by a random occurrence of data. Thus, for a reliable consensus set of size g, we expect the probability $P(n_a \geq g)$ to be low, meaning that it is highly unlikely to be an ACS.

Below, we will derive explicit expression for $P(n_a \geq g)$. To simplify notations in the remainder of this section, we restrict ourselves to transformations of the form

$$\begin{cases} x' & = a_1 x + a_3 \\ y' & = a_5 y + a_6 \end{cases}, \tag{8.3}$$

and square GCRs and CURs, respectively of height τ and ϵ. Note that the restriction is not essential from the theoretical viewpoint, but enables us to simplify notations, and to reduce the dimensionality of the search space. The presented results can be extended to more general transformations in a straightforward manner.

8.3.2 Estimate the Confidence Measure from a Single Correspondence Pair

Our aim is to verify which consensus measure values cannot occur by accident in a given data distribution, i.e., which consensus cannot be ex-

plained by a random distribution of the features over the image. Therefore, we first have to model the random situation by a uniform feature distribution over the image.

Now, suppose we are given a sample set consisting of a single correspondence pair, $\{(\mathbf{x_i}, R'_{ij})\}$, what is the probability that an ACS of some size occurs? That is, how likely is a consensus measure $n_a = n(ACS(\mathcal{T}_{ij}))$ for the CTUP $\mathcal{T}_{ij} = \mathcal{T}(\mathbf{x_i}, R'_{ij})$ for a random feature distribution?

The value of n_a will depend strongly on the size of the IURs. If the CURs are given as square regions, the IURs for a TUP $\mathcal{T}_{ij}$ with transformations of the form (8.3) are again squares whose size varies over the image (see section 5.4.2). The actual width of the IURs varies linearly between ϵ and τ for square GCRs of size τ. Then we assume that the average height of the IURs of $\mathcal{T}_{ij} = \mathcal{T}(\mathbf{x_i}, R'_{ij})$ is given by

$$\delta = \frac{\epsilon + \tau}{2}. \tag{8.4}$$

Let $R'(\mathbf{x_k}, \mathcal{T}_{ij})$ be the IUR of an arbitrary feature point $\mathbf{x_k}$ in the first image. We assume that all feature points in both image are distributed randomly over the entire image. Thus, the probability that a feature point in the second image lies in this IUR $R'(\mathbf{x_k}, \mathcal{T}_{ij})$ is equal to the ratio $\delta^2/(wh)$, with w and h respectively the width and the height of the image.

The second image contains N_2 feature points in S'. Hence the probability P_s that for a given point $\mathbf{x_k}$ there is at least one feature point $\mathbf{x'_{kl}}$ in the IUR $R'(\mathbf{x_k}, \mathcal{T}_{ij})$ is equal to

$$P_s = 1 - \left(1 - \frac{\delta^2}{wh}\right)^{N_2}. \tag{8.5}$$

What is the probability $P(g)$ that an $ACS(\mathcal{T}_{ij})$ *accidentally* contains correspondences for g features in S? First, note that $ACS(\mathcal{T}_{ij})$ already contains the pair $(\mathbf{x_i}, \mathbf{x'_{ij}})$, and that the first set S contains N_1 features. For the remaining $N_1 - 1$ points in S, there must be exactly $g - 1$ points that add 1 to the consensus measure n_a. Hence the probability $P(g)$ is given by the binomial distribution

$$P(g) = \binom{N_1 - 1}{g - 1} {P_s}^{g-1} (1 - P_s)^{(N_1 - g)}, \tag{8.6}$$

for $1 \leq g \leq N_1$.

Thus, the probability $P(g)$ depends solely on the sizes of both the GURs and the CURs (ϵ, τ), on the number of feature points (N_1, N_2), on the width (w) and height (h) of the images, and on the value of the consensus measure (g).

The probability that there is an $ACS(\mathcal{T}_{ij})$ of size $n_a \geq g$ is equal to

$$P(n_a \geq g) = 1 - \sum_{0 \leq i \leq g-1} P(i). \tag{8.7}$$

Illustration. Figure 8.11 (a) shows an illustration of the binomial distribution $P(g)$ (solid line) for the parameters $N_1 = 20$, $N_2 = 160$, $\epsilon = 3$ and $\tau = 80$. We see that the probability $P(n_a \geq g)$ (dashed line) of finding an ACS of size greater than g is low (< 0.01) for $g = 15$.

If, during a matching procedure for a *non-random* data distribution, a consensus set $C(\mathcal{T}_{ij})$ of size 15 (or larger) is actually found for some $\mathcal{T}_{ij}$, we are confident that it cannot be explained by a random occurrence of features, but arises from features related by a transformation (which is well approximated by $\mathcal{T}_{ij}$). For another consensus set $C(\mathcal{T}_{kl})$ of size 7, the confidence level is not acceptable, as the probability that it arises from some random distribution of features is too large.

We notice that for increasingly larger sizes of the GCR, the probability distribution shifts to the right. If we can constrain the candidate matches to a relatively small region, e.g., $\tau = 60$, then the probability that $n_a \geq 10$ just by coincidence becomes very small. For larger GCRs, that probability increases significantly.

A similar effect can be seen for increasing ϵ and N_2 in Figure 8.11 $(c - d)$. When ϵ gets larger (in case of a less precise feature detector), or N_2 increases (when there are more candidate matches), then also the probability of finding an ACS of a larger size increases. Thus, under these circumstances we must find larger consensus sets $C(\mathcal{T}_{ij})$ to gain sufficient confidence.

We can conclude that the same level of confidence is obtained for an increasingly smaller actual consensus set

- when the uncertainty about the feature location is smaller;
- when the candidate search region is more reduced;
- or, when there are less false matches to be found.

Additionally, we can reduce the size of the IURs, and thus also increase our confidence by choosing more than one correspondence pair in the sample set.

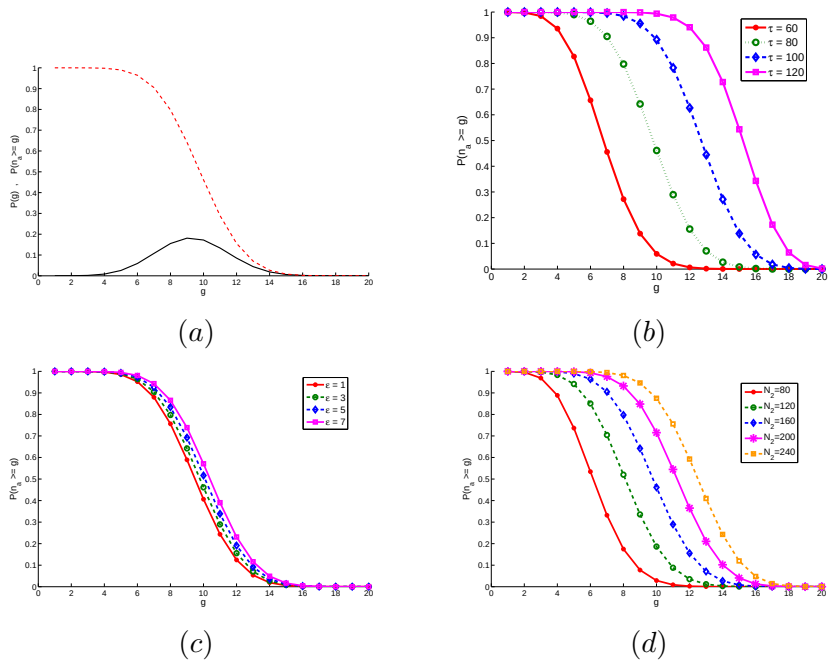


Figure 8.11: Estimated distributions for a correspondence set containing a single pair. (a) The distributions $P(g)$ and $P(n_a \geq g)$ when $N_1 = 20$, $N_2 = 160$, $\epsilon = 3$ and $\tau = 80$. (b, c, d) The effect of increasing values for resp. τ, ϵ, N_2 on the probability $P(n_a \geq g)$, while other parameters remain fixed.

8.3.3 Estimating the Confidence Measure for a Set of Multiple Correspondences

The calculations can easily be extended for sample sets containing multiple correspondence pairs. The major effect is the decreasing size of the IURs. For example, the TUP $\mathcal{T} = \mathcal{T}_{ij} \cap \mathcal{T}_{kl}$ for a sample set containing two pairs, $\{(\mathbf{x_i}, \mathbf{x'_{ij}}), (\mathbf{x_k}, \mathbf{x'_{kl}})\}$ will in general be considerably smaller than the individual CTUPs for a single correspondence pair. Therefore, the IURs $R'_o(\mathbf{x_o}, \mathcal{T}), o \neq i, k$ will also be smaller.

We do not know a precise analytical estimate or upper bound for the average size δ_m of the IURs for larger sets with m pairs. Simulations show that for a sample set of 2 correspondence pairs we always find a value smaller than $\delta_2 = \frac{\epsilon+\tau}{4}$. Note that for a real image it is always possible to compute the exact size of the IURs. Hence, we can obtain a good estimate for δ_m, and for the probability $P_m(g)$ that we

obtain a consensus measure of $n(C(\mathcal{T}_{ij} \cap \ldots \cap \mathcal{T}_{mn}) = g$, for a set of m correspondence pairs.

Now, the probability $P_{s,m}$ that there is at least 1 feature point $\mathbf{x'_{kl}}$ in the IUR $R'(\mathbf{x_k}, \mathcal{T})$ for a point $\mathbf{x_k}$ is equal to

$$P_{s,m} = 1 - \left(1 - \frac{\delta_m^2}{wh}\right)^{N_2}. \tag{8.8}$$

As for the remaining $N_1 - m$ points in S exactly $g - m$ points must add 1 to the consensus measure n_a, we find a binomial distribution,

$$P_m(g) = \binom{N_1 - m}{g - m} {P_{s,m}}^{g-m} (1 - P_{s,m})^{(N_1 - g)}, \tag{8.9}$$

for $m \leq g \leq N_1$.

In this case, the probability that there is an ACS of size $n_a \geq g$ is given by

$$P_m(n_a \geq g) = 1 - \sum_{0 \leq i \leq g-1} P_m(i). \tag{8.10}$$

Illustration. Figure 8.12 shows the probabilities $P_2(g)$ and $P_1(g)$ for the same parameter set ($N_1 = 20$, $N_2 = 160$, $\epsilon = 3$ and $\tau = 80$). Note that the probability of occurrence of an ACS of size g is much smaller when the TUP is computed from 2 correspondence pairs instead of one. For example, it is less likely that an ACS of size 10 (or larger) will occur when starting from 2 correspondence pairs, while this is very likely when computed for 1 correspondence pair.

Practical Example. We consider matching problems like the situation indicated in Figure 8.13 (a), with a rectangular GCR R'_i for each feature $\mathbf{x_i} \in S$ (indicated by the crosses). We find multiple candidate matches $\mathbf{x_{ij}}$ (indicated by dots) in each R'_i. The actual matches related by a transformation are indicated by a solid line. Note that there is a substantial fraction of false candidates present in the feature set.

Figure 8.13 (b) and (d) respectively indicate the probability $P_1(g)$ and $P_2(g)$, computed for a sample set of respectively a single and 2 pairs as explained above for the parameters derived from the situation in (a). The numbers N_1 and N_2 can be related to the number of candidates features in a GCR. We have seen that the feature density has a considerable effect on the distributions.

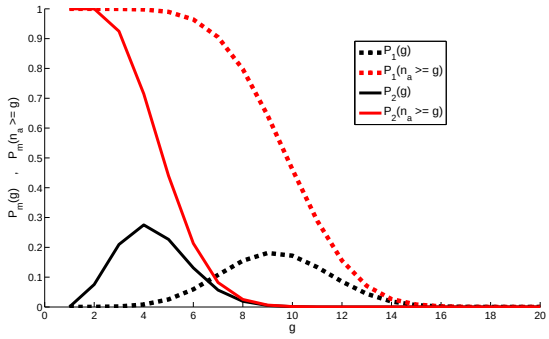


Figure 8.12: The estimated distributions for a correspondence set containing resp. a single pair (dashed line), and two pairs (solid line) when $N_1 = 20$, $N_2 = 160$, $\epsilon = 3$ and $\tau = 80$.

During our matching procedure, we can actually compute all IURs for each $\mathcal{T}_{ij}$. Thus, it is possible to estimate their size more accurately than by using the estimate δ (Eq. 8.4).

If we compute the consensus measure $n(C(\mathcal{T}_{ij}))$ for each $\mathcal{T}_{ij}$, we can construct a frequency histogram of all $n(C(\mathcal{T}_{ij}))$, as shown in Figure 8.13 (c). Notice that the histogram contains larger measures than predicted for ACSs from a random distribution of feature points (b). The graph shows a *bimodal* instead of a binomial distribution (c). It should be bimodal because a subset of features is not located randomly at all. Among the feature points $\mathbf{x'_{ij}} \in S'$ occur also the actual matches for the feature points $\mathbf{x_i} \in S$, i.e. the inliers for the transformation. Thus, the consensus measure $n(C(\mathcal{T}_{ij}))$ for a *correct* all-inlier sample set $\{(\mathbf{x_i}, \mathbf{x'_{ij}})\}$ must be larger than what is expected from a random distribution of feature points.

As a conclusion, the first peak in the bimodal histogram is due to the random feature points that satisfy a binomial distribution, and the second peak is due to feature points that correspond to the transformation. We see that the confidence measure for a given consensus measure can give us some indication about how promising a sample set $\{(\mathbf{x_i}, \mathbf{x'_{ij}})\}$ is, even when it still consists of a single pair.

Figure 8.13 (d) shows the binomial distribution $P_2(g)$ as given by Eq. 8.9 for the situation illustrated in (a), taken over all possible pairs $(\mathcal{T}_{ij}, \mathcal{T}_{kl})$ whose intersection was non-empty. Figure 8.13 (e) shows the actual frequency histogram of the consensus measure $n_a(C(\mathcal{T}_{ij} \cap \mathcal{T}_{kl}))$

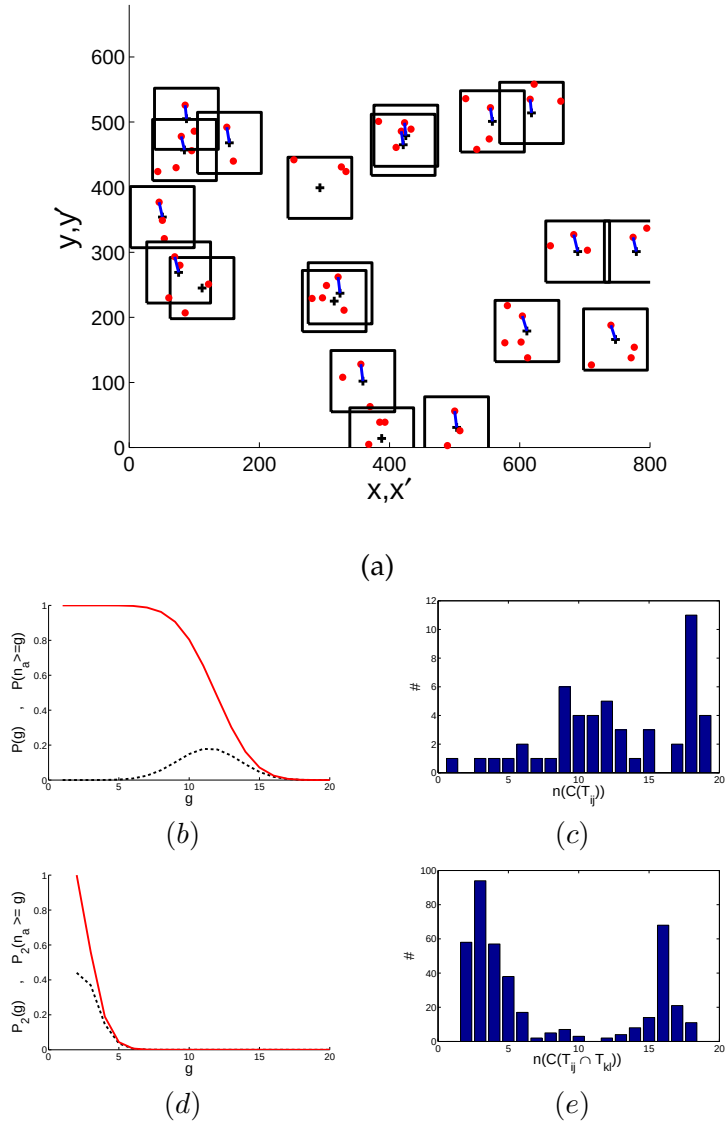


Figure 8.13: Estimated probabilities in a matching example for situation (a). (b) The distributions $P(g)$ (dashed line) and $P(n_a \geq g)$ (solid line). (c) The frequency histogram of all consensus measures $n(C(\mathcal{T}_{ij}))$ for sample sets containing 1 pair. (d) $P_2(g)$ (dashed line) and $P_2(n_a \geq g)$ (solid line) for ACSs starting from two correspondence pairs. (e) The frequency histogram of all consensus measures $n(C(\mathcal{T}_{ij} \cap \mathcal{T}_{kl}))$ for sample sets of 2 pairs.

for sample sets containing 2 pairs. We again see that the probability $P(n_a \geq g)$ gives us a useful indication about how promising a sample set $\{(\mathbf{x_i}, \mathbf{x'_{ij}}), (\mathbf{x_k}, \mathbf{x'_{kl}})\}$ actually is.

If we only consider the *actual* consensus sets obtained for sample sets composed of 2 matches (indicated in Figure 8.13 (a)), we see that these are larger in size than what could occur by coincidence. There are 16 correct matches, for the 20 features in S. The probability that an ACS of this size arose by accident is low (< 0.05) in this case, so that we are confident that the consensus set is reliable. When only 1 sample correspondence pair is chosen, we must find consensus sets of larger size to be sufficiently confident not to have chosen one that arose accidentally. In both cases (i.e. starting from 1 or 2 correspondence pairs), we can distinguish a useful sample set by comparing its consensus measure to the probability of occurrence of an ACS of that size during the construction of the intersection graph. If we ignore sample sets with a low confidence measure, the derivation of the maximal clique in the intersection graph is computationally less demanding.

8.3.4 Conclusion

In a matching problem, we can apply the confidence measure in two ways. First, we can use it as a heuristic to select *interesting* correspondence sets. If the consensus measure for a TUP is likely to be explained by a random distribution of features, we cannot consider this information as reliable and exclude it during the following steps of the matching process. We could for example remove the vertex for that TUP from the intersection graph prior to further processing.

A second approach is to consider the confidence measure when the maximal clique is extracted from an intersection graph. For the CITUP $\mathcal{T}_C = \bigcap \mathcal{T}_{ij}$ for all $\mathcal{T}_{ij}$ in the maximum clique, a CCS is obtained. As a verification step, the result of our matching procedure can be evaluated by the probability $P_m(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ that an ACS of size equal to that of the CCS occurs. If that is unlikely (e.g. the probability $P_m(|\mathrm{ACS}| \geq |\mathrm{CCS}|) < 0.05$), we can accept the CCS as reliable.

We can conclude that our matching framework benefits from this confidence measure. First, we do not need to determine any threshold to be able to distinguish which are the reliable consensus sets. Additional or prior assumptions about the data set are not required, as the confidence measure is estimated from the data itself. Second, the

computations can be sped up, because non interesting samples can be neglected in the construction of an intersection graph. Even more, the confidence in a resulting CCS is larger if we know that it is not at all probable that it arose from a random occurrence of features, even in dense configurations.

8.4 Example Applications

We propose a simple procedure to distinguish correct matching feature pairs and compute the parameters of the transformation. A brief recall: all *correspondence polytopes* are gathered in an *intersection graph*, from which the best matches can be distinguished by computing the *maximum clique*. The actual image transformation can be estimated as the *optimal transformation* computed for the *consistent consensus set* that is obtained from the maximum clique members. We also derived an additional *confidence measure* to verify the reliability of consensus sets.

Our method mainly differs from other techniques by its use of TUPs and intersection graphs to obtain a sufficient level of reliability, even for the small and possibly corrupted data sets. More common approaches typically need a large set of features. Our approach allows to incorporate good, but inaccurately localized samples in the matching process, which is an advantage when working with few features.

After the more theoretical approach in the previous chapters, we will now illustrate that our matching procedure can be a useful tool in image processing applications where typically the geometric relations between features are of importance. We will present our results in a few typical computer vision applications such as image registration and rigid object tracking.

Our feature-based registration procedure exploit the spatial relationship between features to establish correspondences. Feature-based methods show some advantages over intensity-based methods [Zitová and Flusser, 2003], which apply correlation-like techniques to compare intensity patterns in both images. They suffer less from (local or global) intensity changes, do not require interpolation techniques, and support arbitrary transformation types (where most intensity-based methods only suit local translation transformations). We will see that tracking applications pose even more stringent demands on our matching procedure. The process must be fast and robust, and it must be able to cope with illumination and background differences between two images.

8.4.1 Image Registration for Visual Inspection

We will present how our matching procedure is applied in a visual inspection task for a printing system. A camera mounted on the system takes pictures of a sequence of printed objects, and each picture is compared to a reference image. Due to the mechanics of the camera and the lens system, the transformation of the pictures is not precisely known, and it is our job to recover the transformation from the reference image to each of the pictures of the sequence. The visual inspection task involves the registration to a (correct) reference image, so that a quality check and error control can be performed easily and correctly.

We assume that the transformation is composed of translation and anisotropic scaling (Eq. 8.3). The GTUP is defined by bounds on the maximal values of translation and scaling parameters which can be deducted from the mechanics of the system, i.e., $0.9 \leq a_1, a_5 \leq 1.1$, and $-20 \leq a_3, a_6 \leq 20$. There is no prior knowledge about the contents of the pictures or their statistical properties, but we assume that a sufficient number of detectable features is available in the pictures.

Figure 8.14 $(a-b)$ shows an example in which the Harris features [Harris and Stephens, 1988] are detected for both the reference image and a second image, taken from the sequence. We must find a registration transformation that maps the right image on the left image as accurate as possible. We assume that the detector cannot be completely accurate, because of camera vibration and the motion of the printings. A feature point in the second image may still be displaced a few pixels from the location of the corresponding feature point in the reference image, even after adequate scaling and translation.

Figure 8.14 (d) shows the intersection graph for a small set of 12 features (c), and their candidate correspondences in a GCR R_i'. The parameters for the optimal transformation T_{opt} are derived from the correspondence pairs in the CCS from the maximum clique in the graph. During the registration process all pixels of the test image are first transformed by T_{opt} to the reference image space, and then interpolated using nearest neighbor interpolation.

Evaluation. During experiments on different data sets, accurate registration results are achieved with our matching framework. Our method copes well with the image transformations introduced by the printing system. None of the experiments returned an empty CITUP for the maximal clique of the intersection graph. A comparison of our results

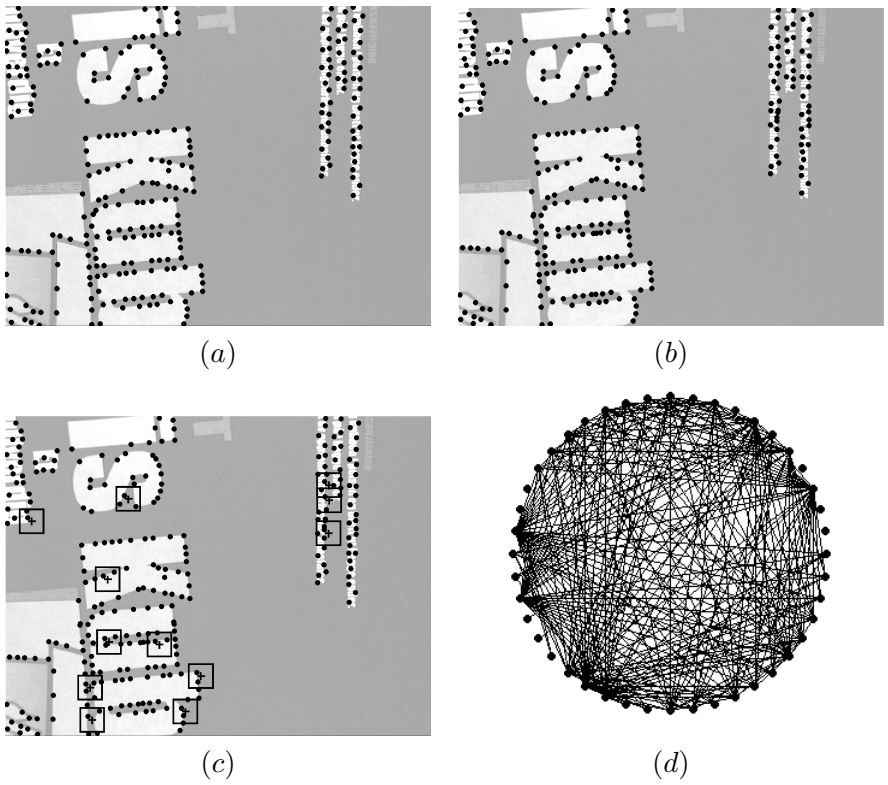


Figure 8.14: Image registration for a visual inspection application (Images courtesy of Dekimo). (a) The detected features in the reference image. (b) Detected features in a test image from the sequence. (c) The GCRs in the test image are centered on a subset of selected feature points $(+)$ from the reference image. Candidate matches $(\bullet)$ must be located within the GCRs. (d) The derived intersection graph.

(UTM) to the results obtained by Dekimo (DHM) is presented in Table 8.2. The details of Dekimo's method cannot be discussed here because of a non-disclosure agreement.

Both methods were evaluated on 4 different image sequences of distinct content for the given number of images in each dataset. The evaluation is conducted based on 2 intensity-based evaluation criteria: a computation of the Mean Absolute Difference (MAD) and the Sum of Squared Differences (SSD) for each registered pair. Table 8.2 shows that similar results are obtained by both methods. The registration transformation obtained by both methods was the same in most cases (certainly

for the first two datasets), although our method requires only very few data points to find the registration transformation.

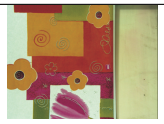
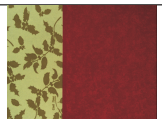
Complexity. Our matching procedure requires a computational effort in proportion to the number of features in the reference image, and the number of candidate matches detected for each feature. The construction of an intersection graph requires the computation of the intersection of CTUPs, which can be done efficiently using linear programming techniques. The derivation of the maximum clique can be enhanced by exploiting the properties of intersection graphs, as explained above (where such algorithms for generic graphs are NP-complete).

As an example, our matching procedure selects 12 features, and derives on average 3 candidate matches for the sketched registration task. Then the matching process requires a computational effort of 1.5 times that of the features detection and extraction step. The registration step demands similar effort (because of the choice for nearest neighbor interpolation).

If we opt to select more features for S the computational effort will of course increase. Experiments have shown that it is possible to correctly derive the CCS for feature sets of 10-25 features with on average 3 candidate correspondences in reasonable time ($< 0.4 - 2s$ in a non-optimized Matlab implementation). If a large number of false positives is extracted (i.e. a higher FD in the GCRs), our matching procedure will run longer, and the success rate will decrease (recall the discussion in section 8.2.6). Therefore, we advise to select only the most pronounced features in the GCR at the risk of excluding the correct match. We could also investigate the selection of candidate correspondence based on appearance (by feature descriptors), at the cost of additional computational effort, but that remains subject for future research.

Conclusion. Our method gives accurate results, comparable to those of an applicable method in an industrial vision task. An advantage of our method is that it is able to yield accurate results for very small sampled feature sets, which is an asset considering computational efficiency. Our matching method benefits from an adequate model for the incorporation of prior information and uncertainty in the transformations, while maintaining robustness. Another advantage is that our procedure allows for a straightforward inclusion of other transformation types, which is more difficult for intensity-based approaches.

Table 8.2: Evaluation of our (UTM) and Dekimo's (DHM) method for the visual inspection application. The first column gives an impression of the image content in the various datasets. The registration of each entry to the reference image is evaluated using Mean Absolute Difference (MAD) and Sum of Squared Differences (SSD).

Dataset		MAD		SSD	
example image	#	UTM	DHM	UTM	DHM
	100	0.0273	0.0273	1201	1201
	100	0.0269	0.0269	1302	1302
	100	0.0208	0.0207	694	691
	100	0.0202	0.0202	559	559
	100	0.0455	0.0455	2794	2819
	100	0.0445	0.0447	2300	2334
	100	0.0408	0.0414	2327	2429
	9	0.0386	0.0384	7266	6859
	7	0.0430	0.0432	4399	4522
	7	0.0529	0.0531	6031	6244
	13	0.0654	0.0641	10771	10086
	13	0.0471	0.0455	7014	5919
	11	0.0438	0.0428	6581	6068
	7	0.0327	0.0330	4972	5185
	7	0.0475	0.0478	5936	6126
	9	0.0435	0.0441	7764	8111
	8	0.0516	0.0503	8914	8210
	13	0.0476	0.0456	10607	9739
	13	0.0388	0.0361	7240	6182
	18	0.0749	0.0827	22839	27281
	13	0.0487	0.0485	10956	10864
	13	0.0428	0.0420	8291	7757
	13	0.0378	0.0388	6800	6992
	12	0.0272	0.0273	4407	4389
	12	0.0230	0.0235	2751	2923
	12	0.0286	0.0286	4032	4018
	13	0.0344	0.0339	5582	5366
	13	0.0522	0.0508	18723	17672
	12	0.0230	0.0231	3357	3381
	12	0.0231	0.0228	3088	2954
	13	0.0491	0.0482	11254	10797
	13	0.0404	0.0413	7616	7822
	13	0.0402	0.0399	6554	6331

8.4.2 Tracking Examples

Our approach for the tracking of rigid objects in video sequences is similar to our above presented registration procedure. That is, find the geometric transformation that best explains the motion of a rigid object from one frame to the next. However, there are some additional difficulties.

Typically very few reliable features are available to describe the object's motion in the image space. Additionally, we must keep track of multiple transformations, i.e., for object features in consecutive frames, and for background items. The computation of different disjoint cliques in the intersection graph should allow to differentiate between several transformations, i.e., one for the background and one (or more) for the moving foreground object(s). This gives an advantage over other robust model fitting methods, which consider only one transformation model at the time.

We consider tracking in different set-ups. First, one or more objects must be tracked in the image sequence from one static camera, where the background is not expected to change significantly. Next, we consider tracking in a single moving camera, where both the rigid objects and the background are non-static. In each described example, we expect that the transformation can be locally approximated as affine, because of the small inter-frame changes in a constrained neighborhood. There are some common issues that require additional attention.

Feature Selection An important problem is the selection of features. How can we discriminate all (moving) objects in the scene? If the camera and the background scene remain static, several approaches are possible. A first approach is to derive which features in frame(t) remain static (i.e. are located at the same location in frame(t+1) albeit with some uncertainty). Second, we can select the features in image regions where motion is detected, e.g., we rely on some simple foreground detection scheme to return the image regions corresponding to moving objects. The choice will depend on application-specific demands.

If the camera is dynamic, the problem is even more complex. In some applications, we can define a region of interest for each object to be tracked. That region of interest can be updated for each frame based on the result of the matching procedure. If such approach is not possible, we could divide or segment the image into multiple regions, and solve the matching problem for each individual region. Eventually,

we could combine the information of neighboring regions. However, this remains subject for future research.

Complexity. For each example, we assume that a sufficient number of features can be detected and tracked on each rigid object, or even in a more constrained region on the object. In the presented tracking tasks we use the Harris detector because of its reliability considering repeatability, and computational efficiency (see section 2.3.2). We do not consider scale invariant features (like SIFT or SURF) because significant scale changes are not expected to occur in between frames for these tracking applications, and because multi-scale detectors are computationally more costly than their single-scale counterparts.

We could opt to select a large set of features in each image region, and try to determine the maximal cliques corresponding to each object and the background from a very large intersection graph. This approach would be computationally very costly. Therefore, we represent each image region by a selection of (the most prominent) features.

We will extract a selection of 10-20 feature points on each object and their candidate correspondences in the consecutive image to construct the intersection graph. According to the results of the experiments in the previous section, this sample set size should be adequate to robustly find a fitting transformation in reasonable time.

Even more, when selecting a larger set of features on each object, we run the risk of extracting less robust or less repeatable features in a typically small image region, hereby introducing a larger amount of false positives and negatives in the matching procedure. Also, the execution time will increase considerably (because the intersection graph would be larger).

In our opinion, it is difficult (if not impossible) to execute our matching procedure in a fixed time span (which could be desirable for some tracking applications) because the number of candidate matches cannot be known in advance. Even for a fixed number of candidates, we cannot predict the number of intersecting CTUPs, which at their turn influence the execution time of the maximum clique algorithms.

Evaluation. We evaluate our matching approach for correspondence problems by checking whether the obtained CCS is indeed correct, i.e., the motion path of an object is described accurately by the transformation derived from the CCS. Additionally, we apply the above presented

measures to verify the reliability of the CCS. Thus, to summarize, for each pair of consecutive frames,

- verify the correctness of the transformation derived from the CCS;
- check whether the CITUP (i.e. the intersection of all CTUPs in the maximum clique) is non-empty;
- weight our confidence by the probability that an ACS of the same size of larger than the CCS could occur;
- apply a larger consensus verification set (CVS) of additional features to verify the correctness of the derived transformation. If a sufficient number of features in the CVS is mapped to their correct correspondences, our confidence in the obtained solution again increases.

Tracking Rigid Objects in One Static Camera

Figure 8.15 (a) presents a first example where one static camera observes a parking lot. It shows an exemplary sequence sampled at regular time intervals. We apply our matching approach to keep track of the vehicles driving up and down through the lane shown in the frames.

A first step is the selection of the features to be tracked. We rely on a simple foreground detection method for this example. Features are only selected in those image regions where the corner measure significantly changes over time. As shown in Figure 8.15 (b), the difference in corner strength shows up strongly where motion is expected (i.e. in foreground regions). We choose a sufficiently large foreground neighborhood as a search area for features in frame(t), at the cost of selecting also background features (c). Note that variation in some feature measure is always expected in foreground areas, no matter which detector one uses.

As this neighborhood covers both the object and the background, not only the effect of false positives and negatives plays its role in the transformation estimation. It is vitally important to be able to differentiate between 2 different transformations in the same region, when only a very limited subset of pronounced features is available.

To select the candidate matches in the next frame(t+1), we can determine GCRs R_i' by relating the relative speed of the vehicles in the world scene to the distance traveled by the features in consecutive frames (see section 8.2.1). Another option is to relate the GCR to the transformation obtained for n previous frames.

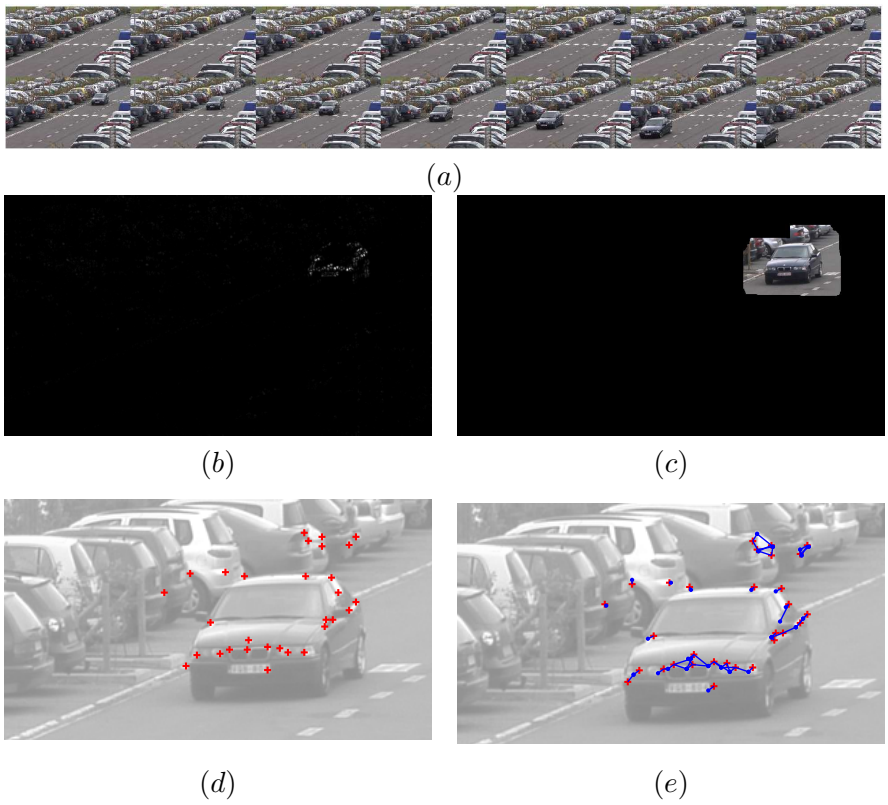


Figure 8.15: (a) Exemplary sequence for the tracking of rigid objects in one static camera. (b) Features are selected in a region where the feature strength measure significantly changes over time. The absolute difference in corner strength is shown for 2 consecutive frames (the larger the difference, the higher the intensity). (c) The extended foreground neighborhood. (d) The selected features $\mathbf{x_i}$ in frame(t). (e) The candidate matches $\mathbf{x_{ij}}$ are selected in a GCR R'_i around $\mathbf{x_i}$ in frame(t+1).

Execute the following steps for each pair of consecutive frames.

1. Detect a set $S = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$ of $n = 10 - 20$ features in a foreground region in frame(t) (Figure 8.15 (d));
2. Detect the set $\{\mathbf{x'_{i1}}, \ldots, \mathbf{x'_{im}}\}$ of candidate matches in frame(t+1) for each $\mathbf{x_i} \in S$ in the GCRs R'_i (Figure 8.15 (e));
3. Take the localization uncertainty into account as a square CUR R'_{ij} for all $\mathbf{x'_{ij}}$ of a few pixels high;
4. Construct the intersection graph $G(\mathcal{T}_{ij} \cap \mathcal{T}_{kl})$, find its maximum

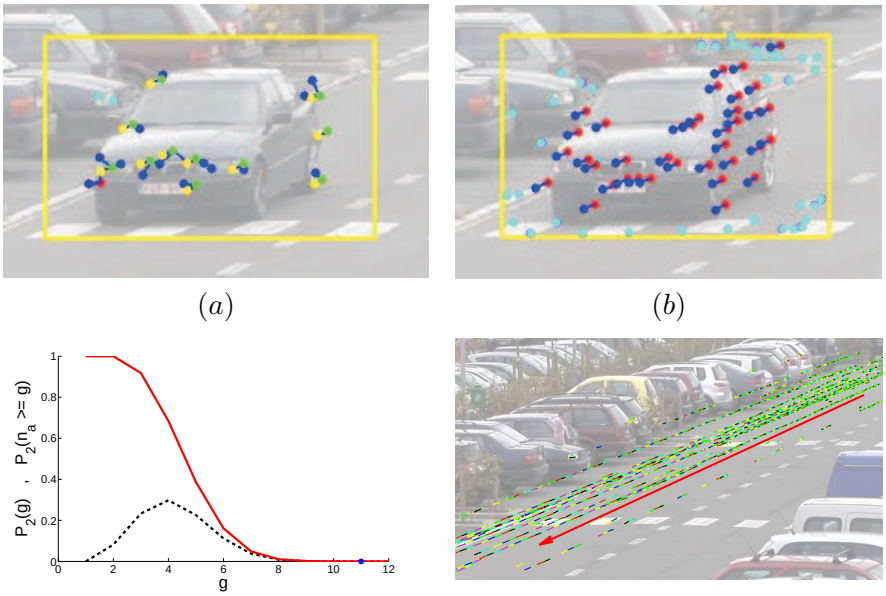


Figure 8.16: (a) The CCS for a non-empty CITUP. The correspondence pairs describe the motion of the vehicle from frame(t) to (t+1) (green-yellow). (b) The CVS of correct correspondence pairs in the verification set S_o (red-blue). Notice that the correspondences located on the vehicle can be differentiated from the background correspondence pairs (cyan). (c) The probability of an ACS occurring for the data in (a) when computed for sample sets of 2 pairs ($P_2(g)$ - dashed line, $P_2(n_a \geq g)$ - solid line). (d) A comparison of the motion described by connecting consistent correspondences in consecutive frames (solid line segments) to the expected motion path.

clique, compute a CITUP $\mathcal{T}_C = \bigcap \mathcal{T}_{ij}$ for all $\mathcal{T}_{ij}$ in the maximum clique and determine the CCS (Figure 8.16 (a));

5. Use a verification set of other features (not in the initial set) to check whether the model is correct (Figure 8.16 (b));
6. (If necessary (e.g. to determine background correspondences, or matches for objects with a different motion path), remove the classes of the r-partite intersection graph that already contain the CTUPs of the maximum clique, and repeat steps 4-5.)

Evaluation. Table 8.3 summarizes the results for our tracking procedure for different camera set-ups in different scenes. We have chosen

Table 8.3: An evaluation of the tracking results for different sequences, illustrated by an example frame in the first column. **A**: Occurrence of a non-empty CITUP (percentage of # frames in the sequence). **B**: Is the motion path correctly described by the CCS (percentage of # frames). **C**: The confidence measure ($P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ averaged over all frames). **D**: The consensus measure of the CCS relative to the number of features selected in frame(t) ($|\mathrm{CCS}|/|S|$ averaged over all frames). **E**: The fraction of correct correspondences in the verification set ($|\mathrm{CVS}|/|S_o|$ averaged over all frames).

Dataset	A	B	C	D	E
	100	100	0.0003	0.88	0.57
	96	96	0.15	0.60	0.77
	92	95	0.005	0.76	0.45

some representative example sequences where the motion path of the vehicle in front of the static camera is well-known, so that we can adequately verify the results. Table 8.3 presents the results for 3 sequences, which are illustrated by an example frame in the first column. Column **A** indicates how often the CITUP was non-empty (in percentage of the total number of frames). Empty CITUPs rarely occur, and if it occurs it can be easily detected. Figure 8.16 (a) illustrates a case where the resulting CITUP was non-empty.

The motion path of the vehicles in these sequences is known, so we can verify whether the resulting correspondence pairs in the CCS are consistent with that motion path. Column **B** indicates the percentage of frames where the correspondence pairs correctly describe the motion path. In over 95% of the frames the CCS lies sufficiently close to the expected motion path. In Figure 8.16 (d), we illustrate the comparison of the connected corresponding features in the CCSs of consecutive frames to the expected motion path for the first sequence in Table 8.3.

Column **C** indicates the average value of $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ over all frames of the sequence, so we can verify whether the confidence measure indeed is a useful indication for the reliability of the CCS. We compute the confidence measure for this example by Eq. 8.9 because the maximum cliques are computed in an intersection graph where each

node reflects the intersection of 2 CTUPs.

As the average value is not adequate to evaluate specific results, an indication of two possible cases is given. First, Figure 8.16 (c) gives an illustration for the data distribution in (a). In this case, the probability $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ that an ACS of the same size as or larger than the CCS (= 11) occurs, is $4\ 10^{-7}$. Thus, the detected CCS is highly unlikely to occur accidentally, and we accept it as reliable. Even in an intersection graph for individual CTUPs, our confidence level would be sufficiently high to accept the CCS as reliable, as the probability $P_1(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ is then only $98\ 10^{-4}$.

Second, Figure 8.17 (a) gives an example of a case where the CITUP is empty. The CCS (of size 4) does not describe the expected motion path (d) correctly. As $P_2(|\mathrm{ACS}| \geq 4)$ is 0.89 for this particular case (c), we are not at all confident in this result. Thus, we must think of another option to derive a correct transformation model. A first solution is to choose another sample subset of features in frame(t) and repeat the process. We can also apply the transformation previously found for the frames (t-1,t) for the current frame pair (t,t+1).

Column **D** indicates the average size of the CCS with respect to the number of features in S. The CCS for the second sequence is on the average smaller than that of the first sequence. A first reason is the higher number of background features selected in sequence 2. Second, there is also a larger number of FN and FP, i.e., respectively the features in S for which there is no correct correspondence in S', and the false candidate matches in S'. This is reflected both in the matching results and in the verification measures, which are slightly worse for this sequence.

We have visualized the reliability measures in Figure 8.18 for (part of) the first 2 sequences in Table 8.3. For the frames where CCS is not correct, $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ rises significantly. There are several frames where the confidence level indicates that the CCS is not reliable, e.g., for the frame pair illustrated in Figure 8.17. Notice also that on average, the probability $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ is considerably higher for the second sequence. First, because of the occurrence of several incorrect CCSs returned by the matching procedure, and second, because of the higher number of FP and FN in the feature sets extracted from the frames of this sequence (see Figure 8.11 (d)).

Therefore, we will not set a hard threshold on the confidence measure, but rather look at the trend for $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ over time. If the probability $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ suddenly increases at time t, we

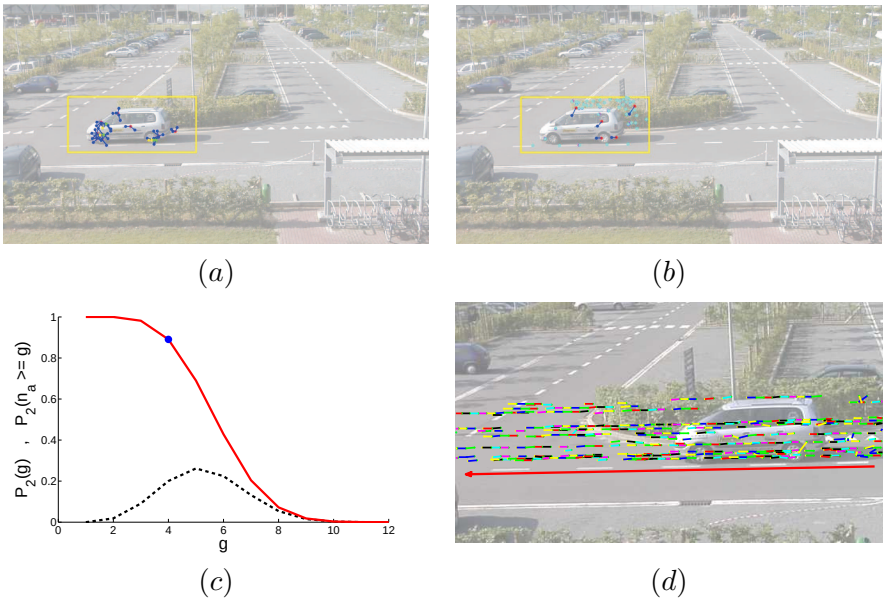
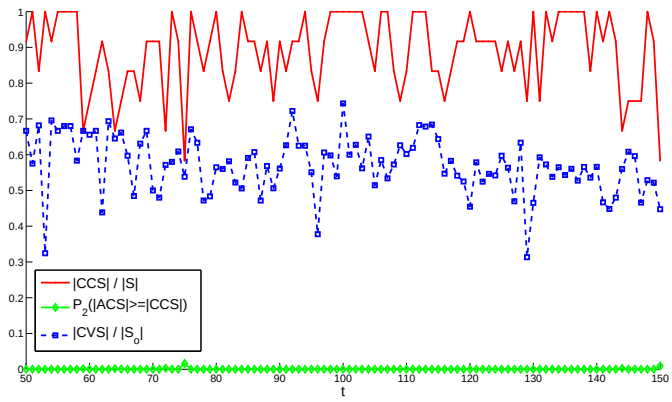


Figure 8.17: (a) The CCS (yellow-green) for a case where both the CITUP is empty, and the confidence level is unacceptable. (b) The set of correct correspondence pairs CVS in the verification set S_o. (c) The probability of an ACS for the data in (a), when the intersection graph is computed for the intersection of every two CTUPs ($P_2(g)$ - dashed line, $P_2(n_a \geq g)$ - solid line). (d) A comparison of the motion described by connecting consistent correspondences of consecutive frames (solid lines) to the expected motion path.

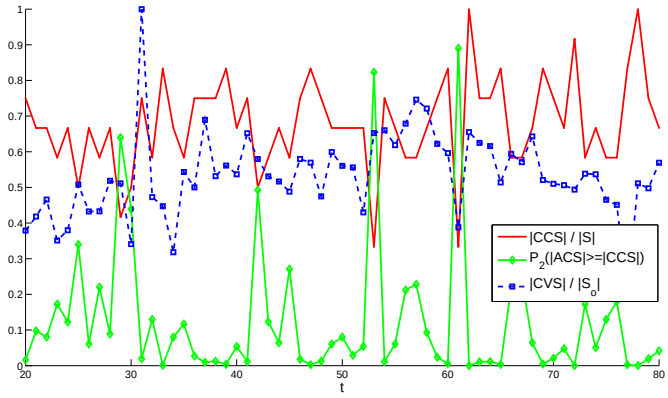
consider the result of the matching process as unreliable.

As a final evaluation step, we consider a verification set S_o of m features in frame(t), not in S. We compute the CITUP $\mathcal{T}_C$ from the CCS, determine the IURs $\mathcal{R}_o(S_o, \mathcal{T}_C)$, and verify whether there is a correct corresponding feature in the set S'_o of m features in frame(t+1). The ratio of the number of correctly matched features $|\mathrm{CVS}|$ to the total number of features in the verification set $|S_o|$ is again a useful indication for the robustness of the matching procedure. This measure is averaged over all frames and indicated in column **E** of Table 8.3.

In the example of Figure 8.16, there is a correspondence found in frame(t+1) for 60% of features in the verification set S_o of frame(t). The number of correctly matched features in the verification set is on average 57% for this sequence, which we consider high with regard to the fact that a large number of features is selected on a small image area.



(a)



(b)

Figure 8.18: Evaluation of reliability measures for tracking applications with a static camera. (a, b) resp. the first and second sequence in Table 8.3.

For large feature density in small image regions, the robustness and repeatability of features decreases because also less discriminative features are selected. Therefore, the correct correspondence may not be present in the set S'_o of m features (as discussed in section 2.4).

When the CCS is incorrect, the number of correctly matched features in the verification set will also be small, as shown in Figure 8.17. Notice that the overall number of matches in the verification set still is high for these frames in Figure 8.18 (b), because we do not distinguish between correspondences found for the foreground object and in the background in that graph.

Conclusion. In our opinion, the results of our matching and tracking procedure are reliable and accurate for these applications. The motion path of an object can be successfully reconstructed based on the information of very small feature sets (i.e. 12 in all examples) and their candidate correspondences in the GCRs. Mostly, the CITUP is non-empty, thus, we can proceed as planned. If it is empty, we follow our previously stated suggestions.

The confidence measure proves to give a good indication about the reliability of the obtained results. We suggest to not set a fixed threshold on this measure, but instead monitor the temporal trend. For example, a sudden large increase for the probability $P_m(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ will indicate the lesser reliability of the CCS for frame(t). If additionally $|\mathrm{CCS}|/|S|$ and $|\mathrm{CVS}|/|S_o|$ decrease, do not retain the obtained result.

It is not straightforward to quantitatively compare our tracking results to those obtained for the artificial examples, as we now compare the CCS to the known motion path, and not to a known set of ground truth correspondences. However, our impression is that the results for real problems are slightly better for comparable parameter sets.

The CCS is correctly extracted for most frames of the example sequences. The fraction $|\mathrm{CCS}|/|S|$ is also quite high, certainly when considering that S can also contain background information, or features on the outline of the objects, for which there are no correspondences described by the motion path. Thus, we conclude that our matching procedure gives satisfying results, considering the limited amount of (possibly corrupted) information that is used.

Tracking by Line Features

Our procedure should also be able to use straight lines (next to points) as features for such applications. Lines offer the advantage of a more stable and repeatable detection under different circumstances. Even more, fewer lines occur as a false positive or negative match in another image, when compared to the situation for point features.

On the downside, there are typically even fewer lines than points detected on an object. The (small) line correspondence set can be used as such to obtain an estimate of the transformation, or the estimated model could serve as a first guess to match point features. The best approach is to combine both in the matching process.

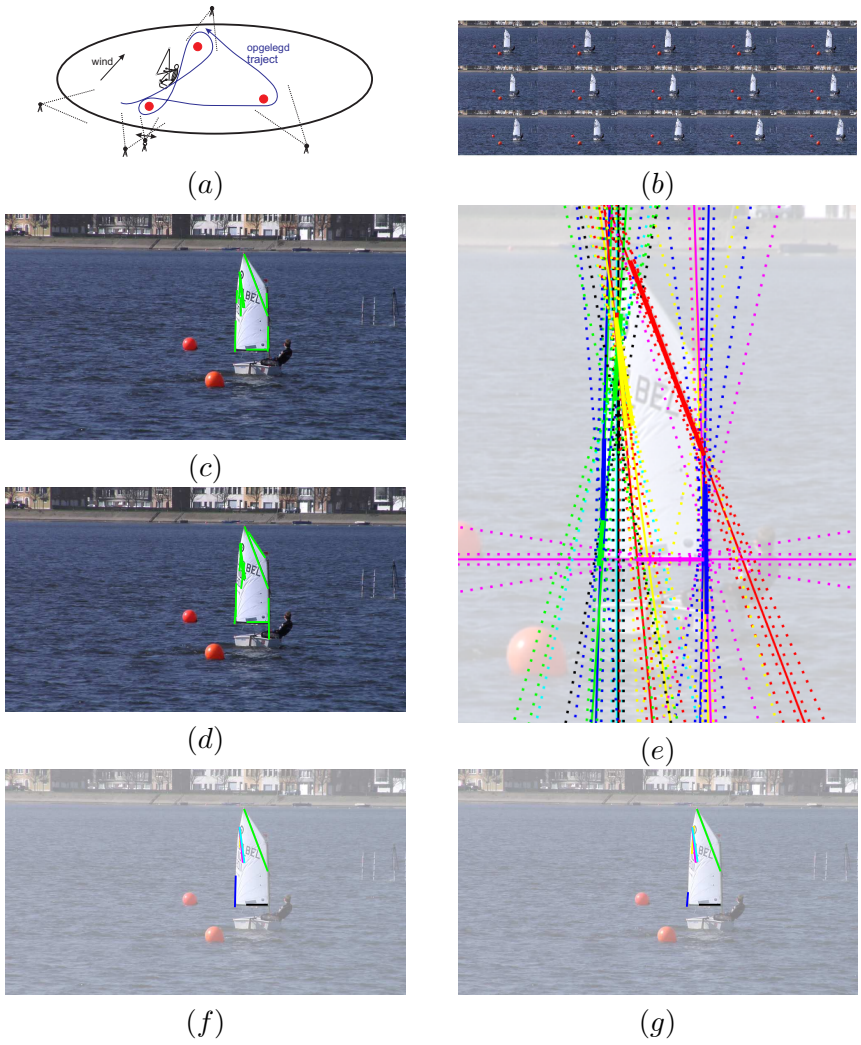


Figure 8.19: $(a-b)$ An illustration of the scene and some example frames in the sequence. $(c-d)$ The line sets detected by our constructive fitting approach [Veelaert and Teelen, 2006b] in two consecutive frames(t) and (t+1) (solid lines). (e) The GCRs in the image space are indicated by dashed lines for frame(t+1). The candidate matches are shown as solid lines. $(f-g)$ The correspondence pairs in the CCS are denoted by solid lines of the same color.

As a short illustration we consider the example sequence presented in Figure 8.19 $(a-b)$, showing a sailing boat as observed by a static camera at the bank. The objective for this application is to automatically track both the sailer and the boat, and relate the effect of the sailer's motion to that of the boat [Callewaert et al., 2009]. For example, the angle between the sailer's rump and the mast is an important indicator for the performance of the sailor as a reaction to weather conditions, wind direction, etc.

We presented the results of point feature tracking for this application in Table 8.3, but in our opinion lines give a better description of the sail structure (as illustrated in Figure 8.19 $(b-c)$) and are more straightforward to use in the athlete performance evaluation.

Our matching procedure requires that we first define a GTUP as a first estimate of the transformation relating the line sets in consecutive frames(t,t+1). We expect that constraints can be derived from the motion of the boat in the world scene. The sailing boat is not expected to change direction quickly, and is sailing away from the camera. Therefore, we consider a GCR which allows limited translation, scaling and rotation in the image space for candidate matching lines in frame(t+1). The GCRs are visualized in Figure 8.19 (d) as dashed lines, with an indication of the candidate matches as solid lines.

Our matching procedure determines a CCS based on parametric consistency in an intersection graph. The resulting matches are given in Figure 8.19 $(e-f)$. The CCS describes the motion path correctly for over 95% of all 200 frame pairs of the sequence.

The CITUP derived from the CCS can serve as an initial estimate for the transformation. To compute the matching transformation more accurately, we can use the CITUP to solve the RoI problem for a set of point features. The estimate of the transformation can then be refined by determining more point correspondences. When using the feature points in frame(t) as a verification set, a large fraction of correct matches is found in frame(t+1) when the CCS describes the motion path correctly.

Tracking Rigid Objects in a Single Dynamic Camera

When both the camera and the objects in its field of view are moving, then our matching procedure must cope with additional problems like the non-static background.

A Constrained Example. The main objective in this application is the accurate evaluation of the motion of the rowers and the oars, and its influence on the propulsion of the boat. The first requirement is the tracking of the rowing boat in a sequence acquired from a moving car, a set-up shown in Figure 8.20 $(a-b)$. Therefore, the few available features on the boat must be matched reliably for consecutive frames so that the position of the boat can be registered for all frames in the sequence, and the rowers' evaluation can proceed.

The second purpose will be the tracking of features in the background scenery, which could return the input for a subsequent ego-localization algorithm for the camera in the world scene. Ego-localization involves the accurate positioning of the camera in the world by extracting the required image information, allowing us to derive parameters such as the speed of the camera, and of the boat.

The task at hand is quite complicated despite the (at first sight) relative simplicity of the overall camera motion. It is difficult to keep the camera steady, when using a larger focal length at such large distance. In this application it is straightforward to compute GCRs by modeling the environment. The rowing boat and the background are (and remain) in different parts of the image, which can be modeled accurately as different planes in the 3D world, as illustrated in Figure 8.20 (c). Then (d) shows that narrow and accurate GCRs can be derived from knowledge about the 3D properties of the scene.

An additional difficulty is that the background consists mainly of trees and grass. Because of the pronounced texture a relatively high number of features is detected in those image regions. Their location is not that reliable or repeatable in consecutive frames (taking the camera motion into account), in our opinion because of varying illumination, shadows and motion blur effects.

Evaluation. As there are very few ($|S| = 20 - 30$ per frame) reliable features detected on the rowing boat itself, the whole set is used as input for our matching procedure. Figure 8.21 (a) shows the features and their candidate matches. Notice that there are relatively few false candidate matches available.

The background features are not adequately positioned if detected as such in the background RoI. Therefore, we look for more reliable and accurately localized features. In this example, we try to use higher-level information. Features are required to be located on the lighting poles,

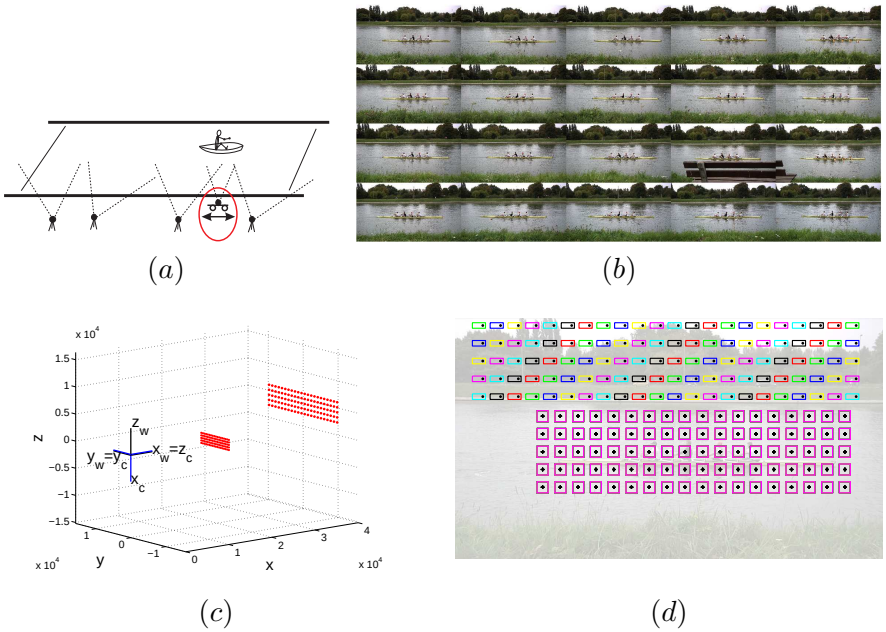
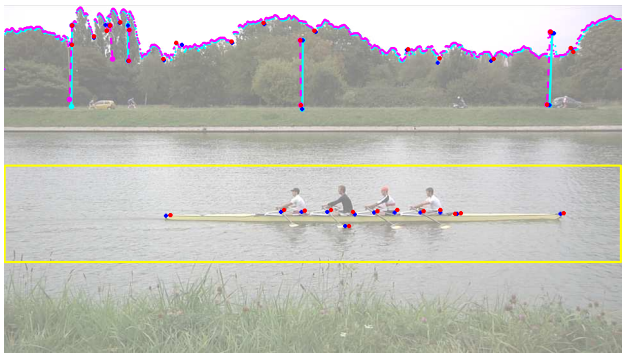


Figure 8.20: (a, b) A situation sketch and example frames of the sequence for the rowing boat tracking application. (c) We construct a pinhole camera model and simplify the 3D properties of the scene as 2 parallel planes. We assume that both the background and the action of the rowers can be captured in a plane parallel to the camera image plane. (d) The GCRs in frame(t+1) for features(+) in frame(t) are derived from the expected motion of the rowing boat and the background scene in the image.

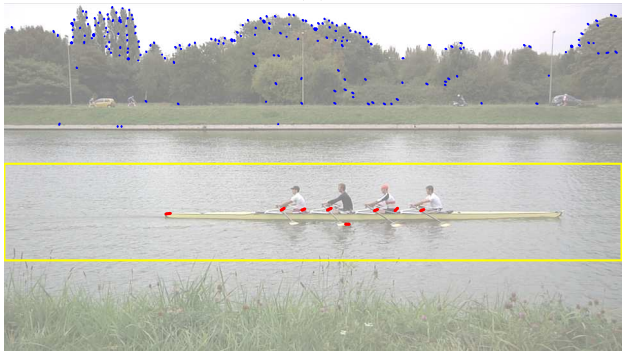
and on the shape of the horizon, as shown in Figure 8.21 (a). The poles can be detected as straight and almost vertical lines in the background. The horizon is represented as a curve from which feature points are extracted where a curvature measure exceeds some threshold.

We proceed along the same steps as given in the description 8.4.2 of the previous section. To evaluate the results, we cannot make use of a simple and clear motion path for the rowing boat (or the rowers), because the camera vibration and motion make it difficult to derive a trajectory of all features in the image (without manual indication in each frame). We will rely on an evaluation of the correctness of the CCSs by an expert user.

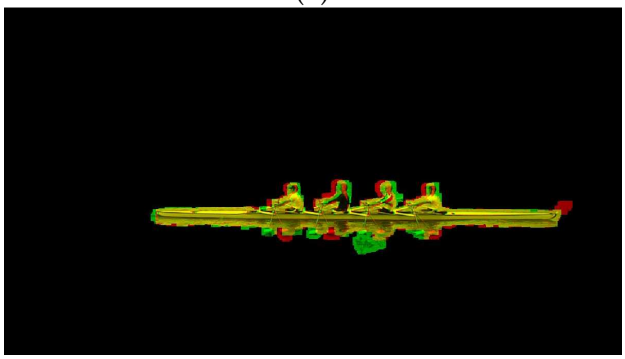
The CITUP is non-empty in 88% of the frames for the foreground sets, and 92% for the background sets. Again we notice that the confi-



(a)



(b)



(c)

Figure 8.21: (a) The detected features in the foreground and background Region of Interest. The features in frame(t) are indicated by crosses and solid lines, those in frame(t+1) by dots and dashed lines. (b) The correspondences in the CCS on the boat, and in the CCS and the CVS for the background. The correspondences on the boat are used to estimate the optimal transformation for the boat from frame(t) to frame(1). (c) The transformed frame(t+1) in green overlay on frame(1) in the red layer.

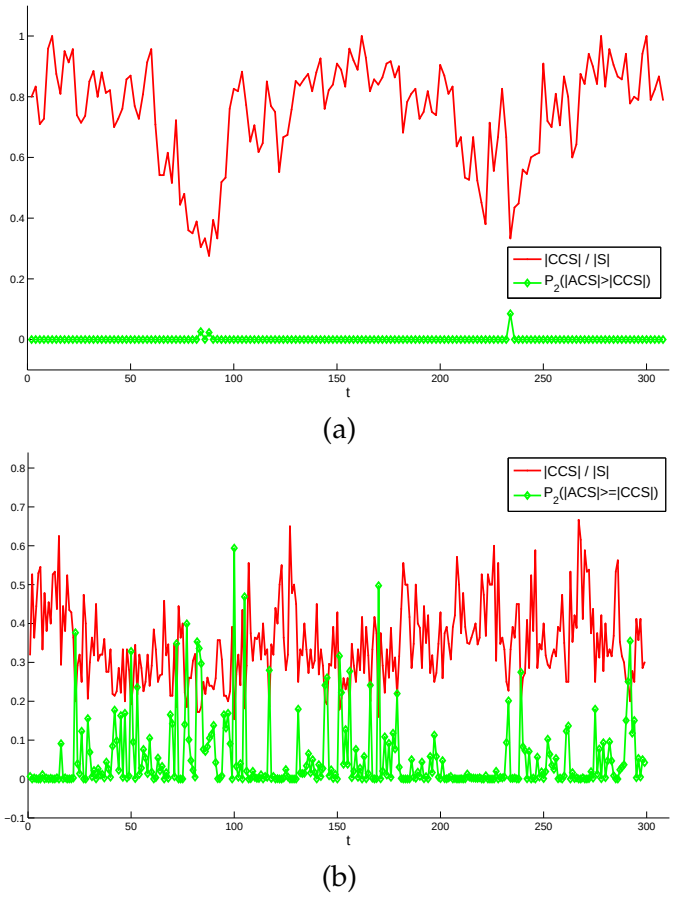


Figure 8.22: Evaluation measures for foreground (a) and background (b).

dence measure gives a useful indication for the reliability of the CCS. Figure 8.22 $(a-b)$ illustrates $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ for resp. the foreground and background. On average, $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ is small for the foreground. There are few false positives, and the resulting CCS is reasonably large in each frame (mostly $|\mathrm{CCS}|/|S|) > 0.5$). For the background, the probability $P_2(|\mathrm{ACS}| \geq |\mathrm{CCS}|)$ is higher on average, because the features are still inaccurately localized, despite our efforts, which results in considerably smaller CCSs. Notice that the probability also suddenly increases, in those frames the CCS is not derived correctly.

Because there are few reliable features in the foreground, we cannot

(a) (b)

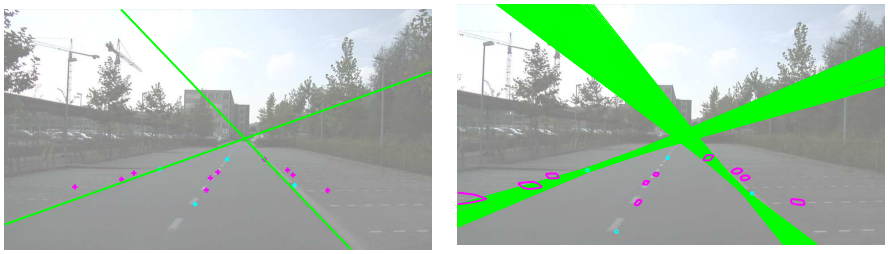
Figure 8.23: Example frames of scenes observed by a dynamic camera mounted on a vehicle.

use an additional verification step by transforming features other than those in the initial set. However, we manage to find a transformation that maps the foreground of frame(t+1) sufficiently accurate to that of frame(t) for most t, as illustrated in Figure 8.21 (c). In this case the boats are registered correctly so that the motion of the rowers is apparent and can be further analyzed.

The General Case for a Dynamic Camera. In general, when the camera is mounted on a vehicle, the 3D scenery is more difficult to model. Some example scene observations are shown in Figure 8.23, A first difficulty are the perspective effects introduced by the 3D structures in the world, that make it difficult to use a simple transformation model for one object, or the background. A second effect that comes into play are the many different textures that occur in the scene, not to mention the changing light conditions, motion blur, etc. These effects make it increasingly difficult to detect sufficiently reliable and repeatable features as the input data, even for our framework.

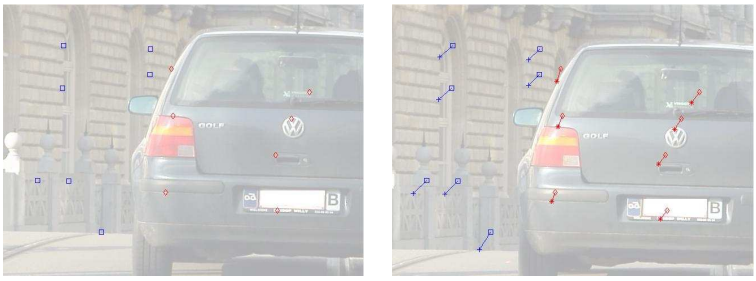
It is possible to track features for a more constrained application, where the features can be detected robustly on the object(s) of interest, e.g., for the rower analysis application.

Another example is sketched in Figure 8.24 where we consider the tracking of features on the road surface, as an essential step in applications such as road mark recognition (e.g. as additional input data for GPS-systems), or for ego-localization. All interesting features for the detection and tracking of road marks can be robustly located in the images of one 3D world plane. Therefore, in this case the transformation



(a) (b)

Figure 8.24: Matched features on one world plane in frame(t) (a) and (t+1) (b).



(a) (b)

Figure 8.25: Matched corresponding features in the frame(t) (a) and (t+1) (b).

can be accurately estimated in our matching framework.

More general problems remain more difficult to solve. Figure 8.25 shows two consecutive frames from a vehicle mounted camera. Applications such as collision avoidance require the detection and tracking of features in the background and on other vehicles.

Our matching procedure is applied to first discriminate the maximum clique in the intersection graph, and its associated CCS. Then, the classes of the graph containing CTUPs in the maximum clique are discarded. Again, a maximum clique is determined for the reduced graph. Figure 8.25 shows two resulting CCSs, where the features on the moving vehicle in front are clearly distinguished from those on the buildings and the road in the background.

This example shows that it is possible in some cases to discriminate the consensus sets in a general 3D scene. However, applications for such scenes introduce several additional difficulties.

- Because of the more complex scene, and the fact that we must estimate multiple different transformations, we must use larger sample sets of features, or derive consistency for smaller image regions and combine the information afterward.
- The size of GCRs must be adapted depending on the speed of the camera/vehicle and the relative motion of other moving vehicles.
- The uncertainty of feature localization will again increase, because of faster varying lighting conditions, motion blur, occlusion, etc.

Future research is required to investigate the capability and robustness of our matching framework in solving such problems. In our opinion, this requires

- an even more robust detection of features, perhaps with the inclusion of more complex features, such as parabolas or circles, or texture information;
- an intelligent selection procedure for features, e.g., where features are collected in subsets that can be designated to higher-level objects;
- the inclusion of appearance information, such as descriptors, also for line segments;
- the inclusion of 3D information for the scene, for instance by estimating the fundamental matrix for consecutive frames.

8.5 Conclusion

This chapter presented our matching procedure by requiring consistency in the transformation parameter domain. The intersection graph implies a first measure for the reliability of a transformation and its consensus set as a combination of consensus measure and implied uncertainty region size. The requirement of finding a consistent consensus set such that the consensus measure is maximal, is fulfilled by computing that set of correspondences for which the largest clique occurs in the intersection graph. By exploiting the properties of such intersection graphs, the maximum clique finding algorithms can be applied efficiently to compute a consistent consensus set.

As an additional confidence measure the consistent consensus size is compared to the probability that a set of that size could arise by a random occurrence of features in an image. These measures each enhance

the probability of finding a correct and accurate consistent consensus set considerably, and as a result the robustness of the proposed matching method increases.

The experiments on artificial and real image data prove that the developed matching framework is sufficiently robust. We have given a choice of example applications to illustrate the robustness of our matching approach. However, a solution for more general and complex situations will be the subject of continued research.

Original contributions. The matching procedure based on finding cliques in an intersection graph for CTUPs is presented in [Teelen and Veelaert, 2005c] and [Veelaert and Teelen, 2006a]. The confidence measure was introduced in [Teelen and Veelaert, 2005a].

Chapter 9

Conclusion

In this final chapter, we will give the general conclusions for the presented work with an indication of future work.

9.1 Overview and Conclusion

In this work we consider a solution for correspondence problems. In chapters 2 to 4, we give an overview of the common approach in computer vision literature at this moment.

The first step of that approach consists of detecting reliable and repeatable features in the distinct images. Most currently developed feature detectors either focus on speed by the design of efficient detection schemes, or on invariance to increase the repeatability rate, or on a combination of both. The most applied detectors are the Harris corner detector (still) [Harris and Stephens, 1988], and the SIFT [Lowe, 2004] and SURF [Bay et al., 2006] methods.

The next step involves an appearance description of the local intensity pattern around the detected locations, commonly referred to as the feature descriptor. Among the most used are again the SIFT [Lowe, 2004] and SURF [Bay et al., 2006] descriptors. All descriptors of the first image must be compared to those of a second, and only the most alike are retained. This results in a large set of putative correspondence pairs. Additionally, one must require spatial consistency for the correspondences in that set. This demand is mostly fulfilled by applying a robust estimation procedure, of which the RANSAC-like methods are the most popular at this time.

The above advocated approach mainly considers large sets of features in two distinct images of related scenes, and estimates one transformation model as accurately as possible. Typically over 1.000 feature descriptors are compared before demanding spatial consistency. This process requires a considerable amount of computation effort. From literature we know that a robust estimation procedure such as RANSAC [Fischler and Bolles, 1981] can cope with up to 50% of false positives in the input data, and requires the tuning of several parameters in advance. The resulting transformation can be accurately estimated from hundreds of correspondences. This approach is mostly used for applications such as image registration, panoramic stitching or object recognition.

Our efforts are focused at correspondence problems where only little and possibly corrupted data is available. We typically select feature sets of up to 30 features in a first image. Among the candidate correspondences in a second image, we expect up to 500% of false positives, i.e., for each correct correspondence for a feature, there are up to 5 false candidate matches. Even more, there can be a considerable amount of false negatives (up to 60%), i.e., features in the first image for which there is no (correct) match in the second image. Furthermore, we take the localization uncertainty for the features into account. The position of a feature can be ill-localized, that is, the location can deviate a few pixels from the expected location (after transformation).

As a solution for such correspondence problems, we present a matching framework that derives a spatially consistent correspondence set by demanding parametric consistency for TUPs, hereby taking positional uncertainty into account. The presented framework for propagating spatial uncertainty can be exploited to describe spatial consistency in terms of the parametric consistency of transformations for a set of correspondence pairs in an intersection graph representation. An additional asset is that prior information about the transformations can be easily included for different applications. Another advantage of the proposed method is that no parameters must be estimated in advance, like in RANSAC-like methods, where e.g., knowledge about the assumed proportion of inliers is required in advance. In our approach, the number of inliers is estimated by the search process itself.

We have applied our methodology to estimate transformations as accurately as possible from small feature sets in different artificial simulations and applications such as registration and tracking. Each building block in our framework was discussed in the consecutive chapters.

In chapter 5, we present a simple mathematical model for representing the localization uncertainty of geometric primitives (points and straight lines), based on the principles of uncertain geometry [Veelaert, 1999b,a, 2001, 2002]. Localization uncertainty is modeled by convex polygonal uncertainty regions in the positional parameter space. This model can be propagated straightforward throughout a geometric reasoning chain. In this work we focused on deriving the uncertainty of transformations from the positional uncertainty.

We model transformation uncertainty as a convex bounded polytope (TUP) in the transformation parameter space, either represented by a set of halfplanes or inequalities, or as the convex span of its vertices. The transformation uncertainty can be reduced by including the information of additional features, or by decreasing the positional uncertainty. The transformation uncertainty is then propagated into the computation of spatial uncertainty for other geometric primitives after transformation. That spatial uncertainty is given as a region of interest (the implied uncertainty region) in the second image. Our model allows for a simple representation of spatial uncertainty and an efficient propagation throughout geometric transformations.

Chapter 6 shows how to incorporate our uncertainty framework into a RANSAC-like robust estimation procedure. Suppose we compute the TUP for a small sample set of correspondences in the hypothesis step. Then, it is important to discriminate in the verification stage which of the features are consistent with the computed TUP, i.e., are in the TUPs consensus set. The incorporation of our uncertainty concepts into the estimation process has several assets. We can naturally include prior information about the transformation in the process. Additionally, taking the propagated uncertainty into account allows for termination of the estimation process after fewer iterations with a correct (spatially) consistent consensus set.

We also show how to compute the one transformation in a consistent TUP that optimally maps the correspondences in the consistent consensus set in the first image onto those in the second image.

Our representation of transformation uncertainty is easily incorporated into different transformation types. In this work we consider both 2D affine and projective transformations (chapters 5 and 7) for point and line features. Part of our future work will consist of extending our uncertainty models toward the perspective transformation and into the computation of the fundamental matrix.

In our opinion, a correspondence problem with small and possibly corrupted data sets requires additional reliability measures. We consider the reliability and confidence for a consistent solution in chapter 8. First, we derive a consistent correspondence set by demanding parametric consistency of TUPs, i.e., there must be a common set of transformation parameters derived from a larger set of correspondence pairs. Our matching method represents the mutual intersection of polytopes in an intersection graph. We apply efficient techniques to find common intersections as (maximal) cliques in the intersection graph.

The computational complexity of our method depends on the number of correspondences. Each correspondence pair adds a number of inequalities to a TUP. Also the size of the intersection graph depends on the number of selected features and the percentage of false candidates. The properties of the correspondence problems are reflected in the construction of the intersection graphs, which allows for a computationally efficient derivation of the maximal cliques. For data sets of up to 40 features in the first image, the computations during the matching problems can be done in reasonable time.

As an additional confidence measure we follow an a contrario reasoning process. If the features would be distributed randomly over the images, what is the probability that the consensus set for a TUP derived from n correspondence pairs can have a certain size? When our matching process returns a consistent correspondence set of size x, we look at the probability that an accidental consensus set of size x or larger could occur in the given data set. If that probability is sufficiently low, we raise the confidence in our result.

Experiments on artificial and real data have shown that our matching procedure can robustly extract the correct solution from heavily corrupted data. For example, for a feature set containing up to 400% of false candidate matches, we find the correct correspondence set in over 90% of the presented examples. Our matching procedure will by definition return a solution that can deviate maximally up to the estimated localization uncertainty from the correct solution. That is the upper bound on the accuracy of the derived optimal transformation in our framework. We conclude that the principles of the developed matching framework can be beneficial in difficult correspondence problems.

9.2 Future

A first and important future consideration is how to design reliable tracking schemes for highly complex scenes by dynamic cameras under varying imaging conditions. As illustrated at the end of the previous chapter, we can now apply our procedure to reliably track points on objects for which we can derive an interesting image region (albeit defined by the application itself, or a foreground region, etc.). More complex and dynamic conditions will require an adequate method for segmenting an image into meaningful regions (individual objects, background, etc.) so that we know where to select features in the image of the 3D scene. A crude segmentation based on texture information could help to distinguish relevant regions [Van Hamme et al., 2008a].

Alternatively, the search space can be reduced by using descriptors for detected features using local intensity information. A combination of these methods can further improve the results. We believe that more robust results can be obtained by including appearance information in the matching process, at the cost of increased complexity. Then, we can exclude many false positives and negatives during the candidate selection step, by ranking a set of putative matches for each feature according to some similarity measure. This requires a lot of additional computational effort, but may offer additional robustness and reliability.

We presented a matching framework that derives a correct correspondence set from little and unreliable information of geometric primitives. We want to extend our uncertainty model toward more complex geometric features such as general conics (ellipses, parabolas and hyperbolas). At this moment, promising results are already obtained by describing complex scenes by parabola segments [Deboeverie et al., 2008], and tracking vehicles over time [Deboeverie et al., 2009]. The description of an object by those higher-level primitives (lines or conics) carries a possible basis for a better semantic description of objects, which in our opinion is beneficial for tracking applications.

Another future path is the extension of our method toward other transformation types, such as in the computation of the fundamental matrix. It should be a straightforward extension, given the current results.

The localization uncertainty does not only affect transformations, but will propagate further along the geometric reasoning chain. For

example, algorithms for self-calibration or ego-localization of cameras based on the information provided by features will suffer from positional uncertainty. For instance, in the rowing application we want to propagate the uncertainty of features located in the background in the recovery of the camera motion path. We think that it is possible to accurately propagate the localization uncertainty far into the reasoning chain.

Appendix A

Adaptive Difference Operators

A.1 Introduction

It is a recurring fact that not only the many feature detectors, but also the feature descriptor methods, are mostly based on the use of differential operators. In fact, almost any algorithm that extracts interesting features from images, such as edges or remarkable points, relies in some way on first or higher order derivatives.

Thus, differential operators are essential in many applications in image processing and computer vision, and they are ubiquitous in low-level vision. A large choice of popular operators exists: simple, like Sobel, Roberts, Prewitt operators, or more elaborate operators that combine Gaussian smoothing with derivatives. For example, the Sobel operator approximates the derivative $\delta/\delta x$ by the difference operator of the shape

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}. \tag{A.1}$$

Figure A.1 shows an example of the application of the Sobel operator on a discrete image.

Some examples of how differential operators prove their worth in feature detection algorithms are SIFT [Lowe, 2004] and SURF [Bay et al., 2006]. SIFT combines the Laplacian with Gaussian smoothing at differ-

Figure A.1: An illustration of the application of the Sobel operator, which is an optimal operator in our framework.

ent scales to extract key points from images. SURF relies on the outcome of digitized box filters, used to approximate Gaussian second order partial derivatives.

We propose to adapt operators to the local image content, i.e., we show how to choose an operator that is optimal for that certain type of local image patch. Our approach is illustrated by an example in LoG edge detection, and the optimality of some widely used filters (such as the Sobel operator) is proven with our framework.

A.2 Image Controlled Difference Operators

Because of their importance for discrete image processing, several approaches have been proposed that either replace a differential operator by a difference operator or estimate its outcome in the context of discrete feature extraction and edge detection applications. Lindeberg [Lindeberg, 1993] shows how to define discrete derivative approximations for low-level feature extraction and edge detection. Gunn [Gunn, 1999] and Demigny and Kamlé [Demigny et al., 1997] consider discrete versions of edge detection algorithms. Lachaud *et al.* [Lachaud et al., 2005] evaluate different accurate estimation methods for the tangent for digitized curves based on digital line recognition.

Remarkably, the difference operators used to approximate a differential operator are often quite rudimentary. The discrete operator is frequently chosen on an experimental basis: for a given class of images, which operator yields the best results visually? One reason is that in the past, even applying a 3x3 linear filter to a large image was an expensive operation. We propose a scheme to select an appropriate operator in a mathematically elegant and profound way, rather than using

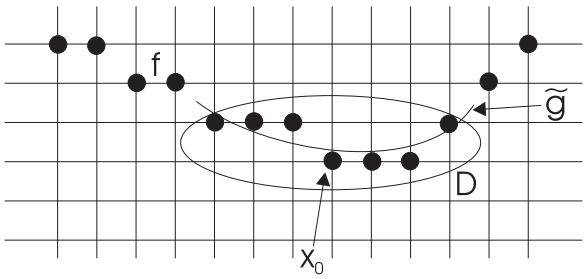


Figure A.2: Finding a tangent to a digitized curve

an ad hoc approach.

Suppose that we want to compute the tangent to a digitized curve in a point p by approximating the curve in a small window around p by some polynomial as in figure A.2. One recurring difficulty is the appropriate choice of the window in which the differential is calculated and the choice of the order of the polynomials. The largest window and the polynomials of the highest order will not necessarily give the best results. If one chooses the order of the polynomial too high, there is a risk of overfitting, while a window chosen too large can result in a degradation of the fit, because the curve may behave differently farther away from p, where, for example, discontinuities may occur. Even more, the fitting of polynomials to discrete curves at discontinuities will yield approximate results. Other influences may come from the appearance of noise in digital images.

We aim at relating the derivation of an optimal difference operator to the analysis of the local image content. Most of the difficulties mentioned above can be solved when the difference operator and the window size are adapted to the neighborhood of each pixel.

In practice, the computation of a derivative in our framework will involve two simple steps. First, a filter bank is used to determine the best class of fitting functions and the optimal window size. Next, a difference operator is used that has been optimized for the specific function class and window size. The design of the filter bank and the optimization of the operators rely heavily on the use of ideals of difference operators. The reduction of difference operators with Gröbner bases will be one of our primary tools to find good operators.

First, we introduce the mathematical concepts and properties involved in our approach. Later, we derive several examples of optimal operators, and introduce a criterion for selecting an appropriate opera-

tor for a local image patch. We will illustrate the proposed operator selection procedure by the computation of an operator that relies heavily on the use of derivative and Laplacian operators on images, the Laplacian of Gaussian (LoG) edge detector. We also show that some widely used operators, such as the Sobel and Prewitt operators, are optimal in our framework and can thus be incorporated in a larger filter bank of optimal, adaptive operators.

A.2.1 Ideals and Gröbner Bases

An ideal I is a subset of elements in a ring R that forms an additive group [Cox et al., 1997]. An ideal I for an arbitrary ring R is defined by the following properties,

- I is a subgroup of the additive group R (for each $x, z \in I$, $x + z \in I$).
- $\forall x \in R$, $xI \subseteq I$ and $Ix \subseteq I$ (if $x \in R$ and $y \in I$, then xy and yx are also in the ideal I);

As an example, the set of even integers is an ideal in the ring of integers. A finitely generated ideal is generated by a finite list of elements $x_1, \ldots, x_n$ and contains all elements of the form $\sum c_i x_i$, where the coefficients c_i are arbitrary elements of the ring R. Also the 0-element of the ring and the ring itself are (trivial) ideals.

In our framework, we want to consider ideals of polynomials $f_i = \sum_{j=0}^{n} p_j x^j$. The polynomial ideal is the ideal generated by a finite set of polynomials $f_1, \ldots, f_n$, defined as

$$\langle f_1, \ldots, f_n \rangle = \{ f | c_1 f_1 + \ldots + c_n f_n, c_i \in R[x] \} \tag{A.2}$$

with $R[x]$ the polynomial ring. If an ideal can be written as in A.2 for a finite set of polynomials, then the ideal is finitely generated. Every polynomial ideal in $R[x]$ is finitely generated (Hilbert Basis Theorem). Now two questions naturally arise. How can one decide whether a polynomial belongs to a given ideal I? And, how can an ideal I be conveniently represented?

A polynomial g is a combination of the polynomials $f_i \in I$ if the remainder of g with respect to I is 0. The division algorithm requires an order of a certain type on the monomials, e.g., lexicographic ordering, where $f_i \succ_l ex f_j$ if the left-most nonzero entry of $f_i - f_j$ is positive, assuming a particular order of variables. For example, for the polynomial ring we have $1 \prec x \prec y \prec x^2 \prec xy \prec y^2 \prec x^3 \prec x^2 y \prec \ldots$.

A monomial ideal is a polynomial ideal that can be generated by monomials. If $x^a \in I$, then all monomials $x^{a+b}, b \geq 0$ are also in I, and one can say that I is closed upwards. A polynomial only belongs to the monomial ideal I if all of its terms are in I.

From now on, we assume a fixed monomial ordering, and denote by $in(f)$ the largest monomial appearing in the polynomial $f = 0$. The initial ideal $in(I)$ of I is the monomial ideal generated by the leading terms of all the elements in I, i.e., $in(I) := \langle in(f) : f \in I \backslash 0 \rangle$.

A finite set of polynomials $g_1, \ldots, g_m \subset I$ is a Gröbner basis for I if the initial ideal of I is generated by the leading terms of the g_i, i.e., $in(I) = \langle in(g_1), \ldots, in(g_m) \rangle$. One can say that a Gröbner basis for a system of polynomials is an equivalence system that possesses some useful properties. Every ideal I has a Gröbner basis G. Furthermore, $I = \langle g_1, \ldots, g_m \rangle$.

The Gröbner basis G enables the algorithmic solution of many problems in computational geometry. For example, the Gröbner bases can be applied to quickly verify ideal membership. One of the advantages of Gröbner bases is that one can reduce systems of difference equations for multidimensional functions, while classical methods are limited mainly to difference equations (or recurrences) for functions in a single variable [Graham et al., 1994]. For an extensive overview of the properties of Gröbner bases, we refer to [Cox et al., 1997; Adams and Loustaunau, 1994].

Using Gröbner bases to solve or reduce difference equations is not new, however. The work by Oberst and Gerdt *et al.*, where Gröbner bases are used to solve systems of partial difference equations (PDEs) has led to renewed interest in the design of efficient discrete schemes for solving PDEs [Oberst, 1990; Oberst and Pauer, 2001; Gerdt et al., 2006]. These new results extend and clarify earlier techniques, such as those found in the work of Collatz [Collatz and William, 1960].

Without doubt, the improved schemes will find their way in image processing where PDEs are used in diffusion processes and image segmentation. The focus of our approach will be on one particular problem: how can we estimate differentials as accurately as possible? We will show that it is possible to compute derivatives more accurately by examining the image locally. In this appendix, we summarize the generalized approach, which was presented in [Veelaert, 1996; Teelen and Veelaert, 2006; Veelaert and Teelen, 2009b], and show how to select the best class of fitting functions as well as the best window size.

A.2.2 Function Classes

To find the derivative of a discrete function f, a standard approach is to approximate f by a continuous function $\tilde{g}$, before taking derivatives. This is a time consuming process, involving unclear choices such as the size of the window for the approximation and the nature of the approximating function. The idea advocated here is that it is not necessary to compute $\tilde{g}$ itself, but that it is sufficient to know by which class of continuous functions the digitized function can be approximated well, and to use a difference operator that is known to be optimal for this class. We introduce a criterion for the selection of an optimal class of approximating functions, and an optimal window size.

We use a continuous real function $\tilde{g} : \mathbb{R}^m \rightarrow \mathbb{R}$ to approximate a digitized function $f : \mathbb{Z}^m \rightarrow \mathbb{Z}$. To approximate the value of a differential operator at a point x_0, it is sufficient to approximate f in a finite subset $D \subset \mathbb{Z}^m$ containing x_0. $|f - \tilde{g}| < \epsilon$ is used as a shorthand for $|f(x) - \tilde{g}(x)| < \epsilon$, $x \in D$. We will introduce classes of approximation functions. Let G be a class of possible approximation functions $\tilde{g}$. Given a domain D, let $G_{\epsilon;D}$ be the set of all $\tilde{g}$ for which $|f - \tilde{g}| \leq \epsilon$.

The shift operator σ^j is defined as $\sigma^j f(x) = f(x + j)$, for $x, j \in \mathbb{Z}^m$. The functional composition of shift operators can be expressed as a multiplication of polynomials, i.e., $\sigma^j \sigma^k f = \sigma^{j+k} f$. A difference operator P can be represented as a polynomial in σ with non-negative, bounded exponents, that is $P = \sum p_j \sigma^j$, and $P \in \mathbb{R}[\sigma]$, the ring of polynomials in σ. We write $P\tilde{g} = 0$ as a shorthand for $\sum p_j \sigma^j \tilde{g}(x) = 0$, $x, j \in \mathbb{Z}^m$. If we write that $|Pf - P\tilde{g}| < \epsilon$, this means that $|Pf(x) - P\tilde{g}(x)| < \epsilon$ for all x for which $Pf(x)$ is well defined, that is $(x + j) \in D$ for every j where $p_j \neq 0$.

Since polynomial ideals are usually defined for polynomials with non-negative exponents, we assume, without loss of generality, that D is a finite neighborhood containing only points with non-negative coordinates.

The difference operators can be represented by templates. For instance, a 2D difference operator $P = \sum p_j \sigma^j = \sum_{j_x, j_y} p_{j_x j_y} \sigma_x^{j_x} \sigma_y^{j_y}$, $j = (j_x, j_y) \in \mathbb{Z}^2$ is represented by a 2D template:

p_{00}	p_{10}	p_{20}	$\ldots$
p_{01}	p_{11}	$\ldots$	

(A.3)

We use the convention that the box at the upper left corner corresponds

to p_{00}. When considering central operators, without loss of generality, we will multiply the operators with σ^{j_c}, with j_c the position of the central point, so that all exponents are non-negative. The same goes for non-central operators, with j_c indicating the point at which the operator is computed. Boxes with vanishing coefficients are either not drawn, or drawn as empty boxes.

Now, we come to the main idea of our approach. Given a pixel in a digital image or point on a digitized curve, how can we select the optimal difference operator? The optimality criterion is built on two different costs, the fitting cost ϵ and the operator cost $|R|$.

The fitting cost. Let L be the differential operator that must be replaced by a difference operator. If we have an appropriate class G of fitting functions $\tilde{g}$, and a difference operator $Q = \sum q_j \sigma^j$ that satisfies $Q\tilde{g} = L\tilde{g}$ for every $\tilde{g} \in G$. Then, if $\tilde{g}$ is an approximation for f such that $|f - \tilde{g}| < \epsilon$ in the domain D, since the operators Q and L are linear, we have

$$|Qf - L\tilde{g}| < \epsilon \sum_j |q_j|, \tag{A.4}$$

which will also be written as $|Qf - L\tilde{g}| < \epsilon|Q|$, where $|Q| = \sum_j |q_j|$.

Hence, the difference operator Q is a good approximation for the differential operator L, provided G contains at least one good approximation $\tilde{g}$ for f.

We will show how we can compute the fitting cost by means of a set of linear filters, i.e., without actually computing a fitting polynomial.

The operator cost. For each class there is an optimal difference operator R, with associated cost $|R|$. We will show that R is optimal in the sense that its associated cost is minimal.

The combined cost criterion. By multiplying the fitting cost ϵ_i by the operator cost $|Q|$ of the corresponding class, we obtain an upper bound on the total error $\tau = \epsilon|Q|$ (as in Eq. A.4). Note that the total cost is due to two successive approximations. The error ϵ arises when f is replaced by some continuous fitting function $\tilde{g}$, a step which cannot be avoided if we want to compute differentials. The error $|Q|$ is due to the replacement of the differential operator L by a difference operator Q, which avoids the explicit computation of the continuous approximation.

The filter bank scheme. The above properties will be exploited as follows in a filter bank scheme, as illustrated in Figure A.3. A bank of linear filters H_i is applied to an image f, as shown in the upper layer

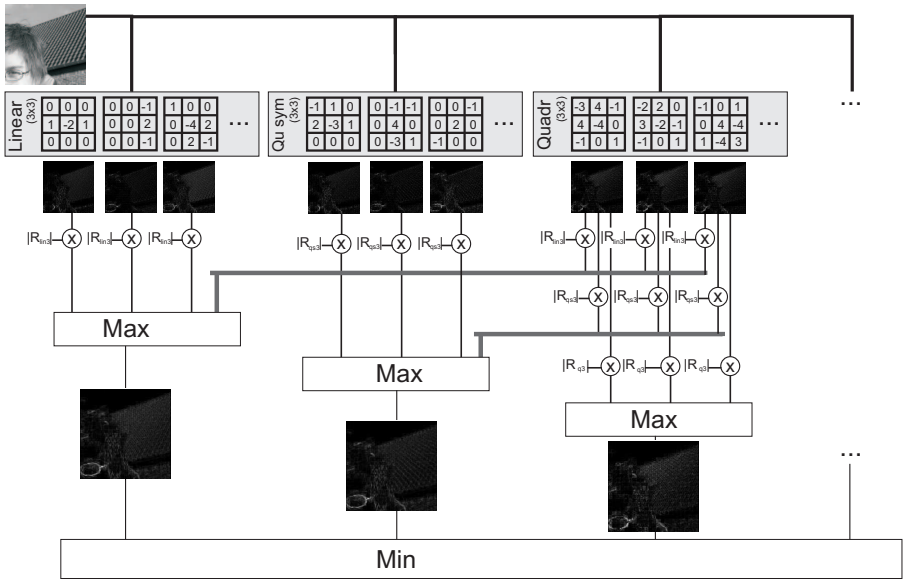


Figure A.3: Scheme of the filter bank for adaptive operator selection.

of the figure. Each filter gives an indication of the maximal error ϵ_i that can be expected when the image f is locally approximated by functions of a given class, e.g., linear or quadratic functions. Since each filter can only estimate the error ϵ_i, more than one filter is needed to obtain reliable estimates for the worst case error. The filters are grouped into subbanks such that each subbank corresponds to a given class of functions.

Then the filter bank can reliably estimate for each function class G_j the maximal error $\tau_j = \epsilon_j |R_j|$ that can be expected by applying the difference operator R_j, that is optimal for that class. The best operator for the local image neighborhood is the one that minimizes the maximal expected error τ_j.

Note that the same filter can often cooperate with other filters to estimate the error for multiple classes (as subbanks may overlap). For example, a function that is locally linear is also locally quadratic, although the difference operator costs may be different.

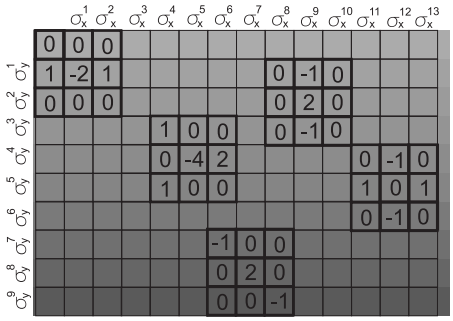


Figure A.4: A class of fitting functions can be defined by means of a set of difference equations $P_i\tilde{g} = 0$. In this example, the operators P_i of the ideal $< \Delta_x^2, \Delta_y^2 >$ define functions $\tilde{g}$ of the form $\alpha_1 xy + \alpha_2 x + \alpha_3 y + \alpha_4$ of class G_5 (see Table A.2). Only a limited subset of operators $P = \sum_i S_i P_i$ from the ideal is shown, as filter kernels on an image. The kernels correspond to five distinct filters in the filter subbank for G_5.

A.2.3 Computing the Fitting Cost

The filter bank scheme only works if two distinct requirements are satisfied. First, the approximation (A.4) is only valid provided $Q\tilde{g} = L\tilde{g}$ holds for $\tilde{g}$. Second, the approximation error ϵ should be as small as possible.

The best way to satisfy both requirements is to define a class of fitting functions by means of a finite set of difference equations $P_i\tilde{g} = 0$, $1 \leq i \leq k$. If a function satisfies these k difference equations, then it will satisfy any difference equation of the form $(S_1P_1 + \cdots + S_kP_k)\tilde{g} = 0$ where the S_i are arbitrary polynomials in the shift operators.

A difference operator P_i can be expressed as $\sigma^h\tilde{g}(x) - \tilde{g}(x) = \tilde{g}(x + h) - \tilde{g}(x) = \Delta_h(\tilde{g}(x))$. For differences in two dimensions, h will be expressed in x or y direction, e.g., $\Delta_x^2(\tilde{g}(x, y)) = (\tilde{g}(x + 1, y) - \tilde{g}(x, y))^2 = \tilde{g}(x+1, y)^2 - 2\tilde{g}(x+1, y)\tilde{g}(x, y) + \tilde{g}(x, y)^2$. That difference can be represented as the operator in the top left corner in Figure A.4. For example, for any function $\tilde{g}$ of the form $\alpha_1 x + \alpha_2$, $\Delta_x^2(\tilde{g}) = 0$, but also $-\Delta_x^2(\tilde{g}) = 0$ or $\Delta_x^3(\tilde{g}) = 0$.

The set of all operators $P = \sum_i S_i P_i$ forms an ideal I, spanned by the polynomials P_i, which is denoted as $I =< P_1, P_2, \ldots, P_k >$. A polynomial ideal I can always be represented by a Gröbner basis. From now on, we will assume that when we write $I =< P_1, P_2, \ldots, P_k >$, then the P_i form a Gröbner basis, according to lexicographic order.

Table A.1: One-dimensional function classes and corresponding Gröbner bases for the defining ideals.

$F_0 : \alpha_0$	$< \Delta_x >$
$F_1 : \alpha_1 x + \alpha_0$	$< \Delta_x^2 >$
$F_2 : \alpha_2 x^2 + \alpha_1 x + \alpha_0$	$< \Delta_x^3 >$
$F_3 : \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \alpha_0$	$< \Delta_x^4 >$
$F_4 : \alpha_4 x^4 + \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \alpha_0$	$< \Delta_x^5 >$
$F_5 : \alpha_5 x^5 + \alpha_4 x^4 + \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \alpha_0$	$< \Delta_x^6 >$

Tables A.1 and A.2 summarize the Gröbner bases for the different classes of respectively one and two dimensional fitting functions. We select polynomial functions of increasing order as function classes F_j for one-dimensional functions, as shown in Table A.1. In this case, the Gröbner bases are simple and can be generated by a single operator. The functions $\tilde{g}$ in class F_1 are defined by $< \Delta_x^2 >$, because all difference operators P_i in that ideal yield $P_i\tilde{g} = 0$, but $< \Delta_x > \tilde{g} \neq 0$.

Figure A.4 shows the templates for a subset of operators P_i in the ideal $< \Delta_x^2, \Delta_y^2 >$, which defines the Gröbner basis for the class of functions of the form $\alpha_1 xy + \alpha_2 x + \alpha_3 y + \alpha_4$.

Advantages. Using an ideal of difference operators to characterize a class of fitting functions has several advantages:

- The finite set of classes of fitting functions form a lattice, i.e., a partially ordered set in which any two elements have a unique supremum or join and an infimum or meet with respect to the union and intersection of the ideals in this set. For example, the structure of the lattice for one dimensional functions is quite simple, since $F_i \subset F_{i+1}$ for each class.
- At each point, the best class of local fitting functions can be determined from the output of a bank of linear filters, i.e., the operators in the ideal. Thus we can design efficient filter banks that compute the approximation error for each class. The elegant structure of the lattice will reappear in the filter bank as the Gröbner bases for the different function classes also form a lattice. This is shown in Figure A.3.
- The optimal difference operator for each function class can be found in a systematic way by using Gröbner bases.

Table A.2: Two-dimensional function classes and their corresponding Gröbner bases (with lexicographic ordering $\Delta_x > \Delta_y$).

$G_1 : \alpha_1$	$< \Delta_x, \Delta_y >$
$G_2 : \alpha_1 x + \alpha_2$	$< \Delta_x^2, \Delta_y >$
$G_3 : \alpha_1 y + \alpha_2$	$< \Delta_y^2, \Delta_x >$
$G_4 : \alpha_1 (x + y) + \alpha_2$	$< \Delta_x - \Delta_y, \Delta_y^2 >$
$G_5 : \alpha_1 x + \alpha_2 y + \alpha_3$	$< \Delta_x^2, \Delta_x \Delta_y, \Delta_y^2 >$
$G_6 : \alpha_1 xy + \alpha_2 x + \alpha_3 y + \alpha_4$	$< \Delta_x^2, \Delta_y^2 >$
$G_7 : \alpha_1 (x + y)^2 + \alpha_2 (x + y) + \alpha_3$	$< \Delta_x - \Delta_y, \Delta_y^3 >$
$G_8 : \alpha_1 (x + y)^2 + \alpha_2 x + \alpha_3 y + \alpha_4$	$< \Delta_x^2 - \Delta_y^2, \Delta_y (\Delta_x - \Delta_y), \Delta_y^3 >$
$G_9 : \alpha_1 (x^2 + y^2) + \alpha_2 x + \alpha_3 y + \alpha_4$	$< \Delta_x^2 - \Delta_y^2, \Delta_x \Delta_y, \Delta_y^3 >$
$G_{10} : \alpha_1 x^2 + \alpha_2 y^2 + \alpha_3 x + \alpha_4 y + \alpha_5$	$< \Delta_x^3, \Delta_x \Delta_y, \Delta_y^3 >$
$G_{11} : \alpha_1 (x^2 + y^2) + \alpha_2 xy + \cdots + \alpha_5$	$< \Delta_x^2 - \Delta_y^2, \Delta_x \Delta_y^2, \Delta_y^3 >$
$G_{12} : \alpha_1 x^2 + \alpha_2 y^2 + \alpha_3 xy + \cdots + \alpha_6$	$< \Delta_x^3, \Delta_x^2 \Delta_y, \Delta_x \Delta_y^2, \Delta_y^3 >$
$G_{13} : \alpha_1 x^3 + \alpha_2 x^2 y + \cdots + \alpha_n$	$< \Delta_x^4, \Delta_x^3 \Delta_y, \Delta_x^2 \Delta_y^2, \Delta_x \Delta_y^3, \Delta_y^4 >$
$G_{14} : \alpha_1 x^4 + \alpha_2 x^3 y + \cdots + \alpha_n$	$< \Delta_x^5, \Delta_x^4 \Delta_y, \Delta_x^3 \Delta_y^2, \Delta_x^2 \Delta_y^3, \Delta_x \Delta_y^4, \Delta_y^5 >$

In the remainder of this section, we will look more closely at each of these advantages.

A.2.4 Lattices of Fitting Classes

Table A.2 shows a possible set of choices for the function classes of two variables. In this case, the classes G_i do not form a chain of subsets as for the one dimensional functions, but rather a lattice $\mathcal{I}$ as shown in Figure A.5. The lattice can be interpreted in terms of function classes as well as ideals. For each pair of ideals I_k, I_l in $\mathcal{I}$ the intersection (or meet) of I_k and I_l, as well as their union (or join or sum) are also included in $\mathcal{I}$. Figure A.5 shows that the function classes G_i form a partially ordered set (or poset) with subset inclusion as ordering relation.

Note that to find a Gröbner basis of an intersection of two ideals, one cannot simply take the union of both Gröbner bases. Instead, one must apply an algorithm for computing ideal intersections, such as the one given in [Cox et al., 1997].

In terms of function classes this means that the set of classes in Table A.2 is closed with respect to intersections and unions. For example, I_{12} is the intersection of I_{10} and I_{11}, or, equivalently, $G_{12} = G_{11} \cup G_{10}$. In this sense, the function classes form a complete collection.

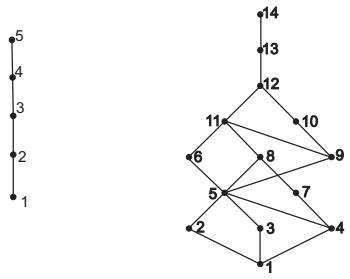


Figure A.5: Lattice of the operator ideals for one and two dimensional function classes shown in Tables A.1 and A.2.

A.2.5 Class Selection with Filter Banks

Instead of actually locally fitting a continuous function, we want to verify whether a class of fitting functions contains at least one member $\tilde{g}$ such that $|f - \tilde{g}| \leq \epsilon$. That is the task of the filter bank. In fact, we have the following immediate result, because a difference operator is a linear operator and $P\tilde{g} = 0$, for all $P \in I$.

Lemma A.1 *If $|f - \tilde{g}| \leq \epsilon$, then the inequality $|f - \tilde{g}| \leq \epsilon$ implies $|Pf| \leq \epsilon|P|$ for each $P \in I$.*

We can prove that the converse is also true [Veelaert, 1996]. Then we can verify whether $|f - \tilde{g}| \leq \epsilon$ by locally computing the outcome of a finite set of difference operators on the image f. The operators can be represented as filter masks H_i, as illustrated in Figure A.4.

Theorem A.2 *There is a finite set of m difference operators $H_1, \ldots, H_m$ in I, such that $|f - \tilde{g}| \leq \epsilon$ holds for at least one member $\tilde{g}$ of the corresponding function class, provided $|H_i f| \leq \epsilon|H_i|$, for $i = 1, \ldots, m$.*

The finite set of operators H_i is computed as follows. Assume that any solution g of the partial difference equations $P_1 g = 0, \ldots, P_n g = 0$ can be written as a linear combination of n functions $g_1, \ldots, g_l : g = \alpha_1 g_1 + \cdots + \alpha_l g_l$. This will be the case when there are enough difference equations to restrict the solution space to a finite dimensional vector space (e.g. by including powers of Δ_x as well as of Δ_y, as in Table A.2).

Let K_D be the set of $\begin{pmatrix} |D| \\ l+1 \end{pmatrix}$ difference operators H_i of the form

$$\begin{vmatrix} g_1(x_1) & \dots & g_l(x_1) & \sigma^{x_1} \\ \dots & & & \\ g_1(x_{l+1}) & \dots & g_l(x_{l+1}) & \sigma^{x_{l+1}} \end{vmatrix} \tag{A.5}$$

where the x_j are $l+1$ arbitrary points in D. The operators of K_D are determinantal expressions of the coefficients $g_j(x_j)$ and the shift operators σ^{x_j}. Since D is finite, K_D is finite. The operators H in K_D are all operators for which Hf is well defined. One can show that $|f - \tilde{g}| \leq \epsilon$ holds, provided $|H_i f| \leq \epsilon |H_i|$ for $H_i \in K_D$ [Veelaert, 1996].

In fact, although up to now, each class of fitting functions was defined by a set of difference operators, one may proceed in the opposite way. With the set K_D, one can find an ideal of defining operators when the solution set is given as a linear combination of basis functions g_i.

When we assume that D is rectangular, there is a unique point x_0 in D with the smallest coordinates. Let $\sigma^{-x_0} K_D$ denote the polynomials of K_D multiplied by σ^{-x_0}. The polynomials $\sigma^{-x_0} K_D$ all have non-negative exponents, but some of the exponents will be zero. One can show that if D is chosen not too small, the polynomials H_i in $\sigma^{-x_0} K_D$ will generate the entire ideal $I =< P_1, \dots, P_n >$. For example, for two-dimensional operators, in practice, it is usually sufficient that D has size at least $(a+2) \times (b+2)$ when the leading monomial of the fitting functions is $x^a y^b$, or smaller.

As stated, for f to be in $G_{\epsilon;D}$, it is sufficient to verify $|Hf| \leq \epsilon |H|$ for all H in the finite set K_D. In other words, it is sufficient to verify a finite set of inequalities to determine whether a discrete function f can be approximated well by at least one function in a given function class. The set K_D is still a very large set, however. One can show that it is sufficient to use only a small subset of operators H_i in K_D to estimate the approximation error ϵ in a reliable way [Veelaert, 1996].

Theorem A.3 *Let $H_1, \dots, H_k$ be a finite subset of the operators in K_D, with $k << m$, such that the operators $H_1, \dots, H_k$ span the entire ideal I. Then there is a positive real number τ, such that if $|H_i f| \leq \epsilon |H_i|$, for $i = 1, \dots, k$, then $|H_i f| \leq \epsilon\tau |H_i|$ for all H_i in K_D.*

τ is close to 1 for a small but well chosen set of operators $H_1, \dots, H_k$. In the filter bank of Figure A.3, each of the operators H_i gives rise to a linear filter.

A.2.6 Computing Optimal Operators

Another benefit of using ideals to define fitting functions is that the optimal difference operator can be selected from a large set of operators in the ideal. The following result follows immediately.

Lemma A.4 *Suppose we have one difference operator Q such that $Q\tilde{g} = L\tilde{g}$, then $R\tilde{g} = L\tilde{g}$ for any R for which $R - Q \in I$.*

In fact, since $P\tilde{g} = 0$ for all operators $P \in I$, if $R - Q \in I$, then $(R - Q)\tilde{g} = 0$, hence $R\tilde{g} = Q\tilde{g} = L\tilde{g}$.

The optimization criterion is that $|R|$ should be as small possible. As a result of Lemma A.4, R can be chosen from an infinite set of difference operators in the ideal of the function class.

The optimal difference operator $R_{G;D}$ for a function class G and a window D is computed by an optimization process. The optimal operator is the operator for which $|R_{G;D}|$ is minimal. We will proceed in two steps. First, we give a general procedure for finding all operators with a predefined shape for a given function class. Then we compute the optimal operator for each class by solving the optimization problem $\min(|R|)$.

Shaped Operators

Difference operators S with a predefined shape have the advantage that known properties of the operator can be taken into account, so that the optimization process can be sped up. For example, if it is known that there exists an optimal symmetric operator, we can use this fact by starting with a predefined symmetrical shape, which reduces the optimization search space considerably.

Given a domain D and an ideal I, defining a function class, a general form for all operators of a predefined shape is derived as follows:

1. Select any difference operator Q, satisfying $Q\tilde{g} = L\tilde{g}$ for this function class, and a predefined shape for the operator S. The shape has zero entries outside the domain D.
2. With the Gröbner basis of I, compute a reduced operator modulo the function class for both Q and S.
3. Identify the reduced predefined operator Q with an operator S of reduced shape, and solve the resulting system of linear equations. If this system has no solution, there is no operator with

the predefined form. When a solution is found, some variables can be eliminated to obtain an expression in which the number of remaining variables equals the number of dimensions of the solution space. The resulting operator represents symbolically all correct operators of the predefined shape.

Example. We illustrate the given procedure by the computation of an optimal difference operator to replace the differential tangent operator $L = \partial/\partial x$ for functions of class $F_2 = \alpha_2 x^2 + \alpha_1 x + \alpha_0$. We can systematically compute a derivative operator by relating the shift operators to partial derivatives so that $L\tilde{g} = Q\tilde{g}$ for any function $\tilde{g}$ in the class. A detailed explanation is given in [Veelaert and Teelen, 2009b].

In this example, L can be correctly replaced by the operator

$$Q = \Delta_x - \Delta_x^2/2 = -3/2 + 2\sigma_x - \sigma_x^2/2 \tag{A.6}$$

for the function class F_2.

As the operator Q is shift invariant, we can shift it by σ_x^2 so that the tangent is computed in the center of a window of length 5, which gives $Q' = -3\sigma_x^2/2 + 2\sigma_x^3 - \sigma_x^4/2$. The operator Q' is then reduced modulo the ideal I_{F_2} of this function class, resulting in $Q' \mod I_{F_2} = 1/2 - 2\sigma_x + 3\sigma_x^2/2$.

The same Gröbner basis is used to reduce the operator S, which we choose to be a difference operator with shape $[-a, -b, c, b, a]$ or $-a\sigma_x - b\sigma_x^2 + c\sigma_x^3 + b\sigma_x^4 + a\sigma_x^5$.

The reduction of S yields $S \mod I_{F_2} = (2a+b) - (8a+4b)\sigma_x + (6a+3b+c)\sigma_x^2$. Note that the Gröbner basis is used here to find conditions for the shaped operator S so that S is the same as Q modulo the ideal. Therefore the monomial order used to compute the Gröbner basis is not important.

Identifying $Q' \mod I_{F_2}$ with $S \mod I_{F_2}$

$$1/2 - 2\sigma_x + 3\sigma_x^2/2 = (2a+b) - (8a+4b)\sigma_x + (6a+3b+c)\sigma_x^2 \tag{A.7}$$

gives a system of 3 linear equations of rank 2:

$$\begin{cases} 2a + b = 1/2 \\ -8a - 4b = -2 \\ 6a + 3b + c = 3/2. \end{cases} \tag{A.8}$$

This system allows us to eliminate 2 variables, resulting in an operator of the general form $[-a, -1/2+2a, 0, 1/2-2a, a]$, where a can be chosen

Table A.3: Optimal tangent operators for the one-dimensional function classes with different window sizes.

size	F_1 & F_2	F_3 & F_4
3	$\frac{-1}{2} + \frac{\sigma^2}{2}$	non existing
5	$\frac{-1}{4} + \frac{\sigma^4}{4}$	$\frac{1}{12} - \frac{2}{3}\sigma + \frac{2}{3}\sigma^3 - \frac{1}{12}\sigma^4$
7	$\frac{-1}{6} + \frac{\sigma^6}{6}$	$\frac{1}{12} - \frac{1}{4}\sigma - \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^4 + \frac{1}{4}\sigma^5 - \frac{1}{12}\sigma^6$
9	$\frac{-1}{8} + \frac{\sigma^8}{8}$	$\frac{1}{24} - \frac{1}{3}\sigma^2 + \frac{1}{3}\sigma^6 - \frac{1}{24}\sigma^8$
11	$\frac{-1}{10} + \frac{\sigma^{10}}{10}$	$\frac{9}{160} - \frac{25}{96}\sigma^2 + \frac{25}{96}\sigma^8 - \frac{9}{160}\sigma^{10}$

freely. Any operator of this form has the required shape, and satisfies $R\tilde{g} = L\tilde{g} = \frac{\partial \tilde{g}}{\partial x}$. We can now minimize the operator cost $|R|$ and obtain the coefficients for the operator R.

Note that Q is not unique. For example $Q = \Delta_x - \Delta_x^2/2 + \Delta_x^3/3$ will also give a correct result. However, the reduction by the Gröbner basis will automatically eliminate unnecessary terms, such as $\Delta_x^3/3$, because the third derivative is always zero for this particular class of fitting functions F_2.

Optimal Operators

An operator is optimal when the sum of the absolute values of its coefficients is minimal. For example, the cost of the operator in the previous example is $|R| = \sum |r_i| = |-a| + |-1/2 + 2a| + |1/2 - 2a| + |a|$.

In fact, although the expression for $|R|$ is non-linear, it is a convex function for which we can easily compute an optimum. In fact, it is easy to see that the minimum is attained where at least one of the absolute values is zero, and that the solution set is a convex polytope. Therefore, the optimum can be found by solving a set of linear programming problems. In this example, we obtain a minimum for cost $|R| = 1/2$, with the template $R = [-1/4, 0, 0, 0, 1/4]$.

Table A.3 shows the optimal tangent operator for function classes of Table A.1, with increasing domain (or window) size. Note that some function classes may share the same optimal operator. For example, this is the case for the functions of first and second order. This is no longer true when the tangent is not computed at the central position of the window.

Table A.4 shows that the operator cost $|R|$ decreases for larger win-

Table A.4: The operator cost $|R|$ of the optimal tangent operators for the one-dimensional function classes, given for different window sizes.

size	F_1 & F_2	F_3 & F_4	F_5
3	1	n. e.	n. e.
5	$\frac{1}{2}$	$\frac{3}{2}$	n. e.
7	$\frac{1}{3}$	$\frac{7}{6}$	$\frac{11}{6}$
9	$\frac{1}{4}$	$\frac{3}{4}$	$\frac{79}{60}$
11	$\frac{1}{5}$	$\frac{19}{30}$	$\frac{153}{140}$

a	b	c	b	-a
b	d	e	-d	b
c	e	f	e	c
b	-d	e	d	b
-a	b	c	b	a

1/8	0	0	0	-1/8
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
-1/8	0	0	0	1/8

Figure A.6: Example: Laplacian operator computation for G_7 with shaped operator (left) and resulting optimal operator (right).

dow sizes in the same function class. When the size of the window remains fixed, we notice that the operator cost increases for higher order function classes. The operator $\frac{1}{12} - \frac{2}{3}\sigma + \frac{2}{3}\sigma^3 - \frac{1}{12}\sigma^4$ is often used in numerical analysis to compute derivatives, although the way in which it was derived here, is completely different from the derivation described in [Henrici, 1964].

Example. The same procedure can be repeated to compute a difference operator to approximate the results for another differential operator, e.g., the Laplacian $L = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ of functions in 2 variables. We illustrate this procedure for function class G_7. For these functions of the form $\tilde{g} = \alpha_1(x+y)^2 + \alpha_2(x+y) + \alpha_3$, L can be replaced by the difference operator $Q = 2\Delta_y^2 = 2\sigma_y^2 - 4\sigma_y + 2$ so that $L\tilde{g} = Q\tilde{g}$.

We choose an operator with the shape given in Figure A.6, compute the linear system and solve the optimization problem, resulting in the operator given on the right in Figure A.6.

If the reduction of the operators with the ideals of different function classes yields operators of the same shape, then the optimal operator of that shape will also be the same for those function classes. Table A.5 shows a table of optimal Laplacian operators for functions in two variables. The set of linear function classes comprises G_1, G_2, G_3, G_4

Table A.5: Optimal Laplacian operators for the 2D function classes in 3x3 or 5x5 templates.

	linear	quadratic symmetric	symmetric	cubic	4th order
3x3	0	1/2 0 -1/4 0 -1/2 0 -1/4 0 1/2	1/2 0 1/2 0 -2 0 1/2 0 1/2	n.e.	n.e.
5x5	0	1/8 0 0 0 -1/8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1/8 0 0 0 1/8	1/8 0 0 0 1/8 0 0 0 0 0 0 0 -1/2 0 0 0 0 0 0 0 1/8 0 0 0 1/8	1/8 0 0 0 1/8 0 0 0 0 0 0 0 -1/2 0 0 0 0 0 0 0 1/8 0 0 0 1/8	-1/24 0 0 0 -1/24 0 2/3 0 2/3 0 0 0 -5/2 0 0 0 2/3 0 2/3 0 -1/24 0 0 0 -1/24

and G_5; the quadratic classes are G_9, G_{10}, G_{11} and G_{12}; the cubic class is G_{13}; the class of fourth order is G_{14}. Although G_6 is not a class of linear functions, applying the Laplacian operator gives the same result as for a linear function. We use the term quadratic symmetric to denote the set of function classes $\{G_7, G_8\}$, which are symmetric in the second degree terms. There are no 3x3 operators that correctly compute the Laplacian for cubics and higher order functions.

The operator cost for all these functions classes in different window sizes is given by Table A.6. For a given window size, the operator cost increases when the function class becomes more general. In fact, Table A.2 shows that $I_{12} \subset I_{10}$, i.e., G_{12} contains G_{10} as a subset, and the functions in G_{10} have to satisfy the difference equations from I_{12} plus some extra difference equations. As a result, the class of operators from which an optimal operator has to be chosen is larger for I_{10} than for I_{12}. Second, for a given function class the cost decreases when the window size increases. In fact, the operators for a larger window contain the operators for a small window as a special case.

Table A.5 shows that, for linear functions, the optimal operator is the zero operator with cost equal to zero. This is correct since the Laplacian of a linear function (and of G_6) is zero, a value which can be estimated without error. This may give rise to some anomalies when we have to select an appropriate function class and window size. In the next section, we discuss how these anomalies can be avoided by introducing an artificial cost for linear functions (shown between brackets in Table A.6).

Table A.6: Cost of optimal Laplacian operators for different sets of function classes and window sizes.

	linear	quadratic symmetric	quadratic	cubic	4th order
3x3	0 (1)	2	4	n.e.	n.e.
5x5	0 (1/4)	1/2	1	1	16/3
7x7	0 (1/9)	2/9	4/9	4/9	9/5

Derivative operators for images

An important application is the computation of derivatives for discrete images. Let

-a	0	a
-b	0	b
-a	0	a

(A.9)

be a proposed shape for a difference operator that computes the derivative $\partial/\partial x$ at its central position. The shape of this operator can be written in terms of shift operators as $-a + a\sigma_x^2 - b\sigma_y + b\sigma_x^2\sigma_y - a\sigma_y^2 + a\sigma_x^2\sigma_y^2$. This operator must be equivalent to the operator $\Delta_x - \Delta_x^2/2 + \Delta_x^3/3 - \Delta_x^4/4 + \cdots$, which computes $\partial/\partial x$.

For constant functions and functions of the form $\alpha_1 y + \alpha_2$, the zero operator is optimal. For all other linear and quadratic functions one finds, after Gröbner reduction of both operators and identifying coefficients, that the 3×3 derivative operator of the proposed shape must have the form

-a	0	a
2a-1/2	0	-2a+1/2
-a	0	a

(A.10)

with operator cost $4|a| + 2|1/2 - 2a|$. The minimal value of the cost is 1, which is obtained for $0 \leq a \leq 1/4$. It is interesting to note that the Prewitt, Sobel and Frei-Chen operators are all optimal in this sense (choose $a = 1/6$, $a = 1/8$, or $a = 1/(2(2+\sqrt{2})) \approx 0.146$, respectively).

In practice this means that if an image can be locally approximated by a polynomial function such that $|i(x,y) - \tilde{g}(x,y)| < \epsilon$, then the worst case error in computing the derivative is minimal when $0 \leq a \leq 1/4$.

$\frac{1}{40} \times$

-1	0	0	0	1
-2	0	0	0	2
-4	0	0	0	4
-2	0	0	0	2
-1	0	0	0	1

$\frac{1}{24} \times$

1	0	0	0	-1
0	-8	0	8	0
0	0	0	0	0
0	-8	0	8	0
1	0	0	0	-1

Figure A.7: Optimal operators in a 5×5 window for linear and quadratic function classes with cost $1/2$ (left), and for cubics with cost $3/2$ (right).

Furthermore, both latter operators exploit the limited freedom of the parameter a to perform some smoothing in the vertical direction.

There is no 3x3 operator that correctly computes the derivative for cubic functions. In fact, it makes no sense to use cubics in a 3x3 window to approximate an image function, since a cubic has 10 parameters, and the fit will always have zero error. In Figure A.7, we show an optimal operator in a 5x5 window for linear and quadratic functions with cost $1/2$, and one for cubics with cost $3/2$.

A.3 Optimal Operator Selection in Practice

The worst case error produced by a difference operator is equal to $\epsilon|R|$. Several parameters affect the quality of the approximation in different ways:

- When the window size increases, the fitting cost ϵ cannot decrease for a given function class.
- When the window size remains fixed, ϵ cannot increase for more general functions (e.g. polynomials of high order).
- When the window size increases, the operator cost $|R|$ cannot increase for a given function class.
- When the window size remains fixed, the operator cost $|R|$ cannot decrease for more general functions.

That is, increasing the window size can have a beneficial as well as bad influence on the approximation, and the same goes for the choice of the function class. When applying our framework to compute a derivative for an image, as in Figure A.3, we need a criterion to select the best operator for each pixel neighborhood. Which function class is the best approximation for the local image surface? And which window size should we consider?

When we take all the above effects at once into account, it makes sense to use $\epsilon|R|$ as the criterion for selecting the best operator. It is the maximal error we can make in the estimation of the derivative, and ϵ can be easily estimated in the image neighborhood by a limited set of filters H_i.

There may be some problems, however, which prohibit the blind application of this criterion to select the optimal operator, adapted to the local image content.

First, some operators have zero cost. As noted in Table A.6 we have $|R| = 0$ for the Laplacian of a linear function, and therefore $\epsilon|R| = 0$.

Second, the fitting error may be zero. Suppose that we compute the tangent of a discrete curve on which three subsequent points are collinear by accident. Then the fitting error ϵ of class F_1 in a window of size 3 will be zero. Therefore, the error $\epsilon|R|$ will be smaller (or equal) for a window of size 3 and an operator of class F_1, than for any other function class or window size. This phenomenon will appear whenever the window size is small compared to the order of the polynomial. For example, three points of an arbitrary function always lie on a parabola, which makes the fitting error zero. Also, four points of a digitized curve will still often lie by accident on a parabola.

There is no obvious solution for these problems since they are related to the loss of information that arises when a curve or function is digitized. How to cope with these artifacts may often depend on the application. For example, the noise level in an image influences the decision to accept that a function is locally linear.

One simple way to avoid these anomalies is to incorporate two additional, artificial costs in the selection criterion: $(\epsilon + \delta_\epsilon)(|R| + \delta_{|R|})$, where δ_ϵ and $\delta_{|R|}$ are small positive real numbers.

Alternatively, one can introduce artificial costs for some of the operators, as was done in Table A.6. For example, where the image is locally best approximated by functions that are linear, the best Laplacian difference operator is the zero operator, which has zero cost. To avoid that this operator is always preferred over the other operators by the selection criterion, we introduce an artificial non-zero cost in the column of the linear operator, which is in line with the costs of the other operators.

By observing Table A.6, one sees that the cost $|R|$ often goes down by a factor 2 when the function class gets more specialized. We shall extrapolate this trend and use a cost for linear functions which is half the cost of symmetric quadratic functions, as shown between brackets

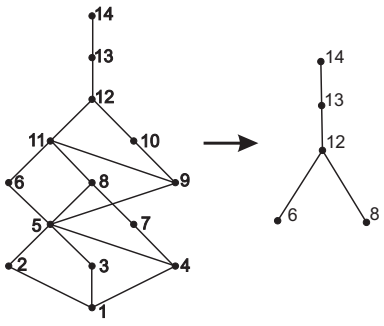


Figure A.8: The lattice of the two dimensional operator ideals shown in Table A.2 on the left is now reduced to the right lattice because of the occurrence of identical optimal Laplacian operators for several subsets of function classes.

in Table A.6. This means that the optimal operator for the quadratic or higher order function classes will only be selected if their fitting cost is at least less than one half the fitting cost of linear functions. By a fortunate coincidence, the cost used in a 3x3 window for linear functions is simply the fitting cost ϵ itself.

A.3.1 Difference Operator Selection for LoG Edge Detection

We will now give an example of how to apply the selection criterion in the computation of the Laplacian for a discrete image. In each pixel, we want to select a difference operator that yields accurate results for the functions that locally best approximates the image. Therefore, we apply a filter bank with a similar setup as presented in Figure A.3. First the maximal fitting cost is computed by a subbank of filters for each class of functions. Then we apply the optimal difference operator for that function class for which the selection criterion $\epsilon|R|$ is minimal.

We have seen that the function classes G_i of Table A.2 form a poset with subset inclusion as ordering relation. When computing optimal operators for all classes by reducing the Laplacian operator $\partial^2/\partial x^2 + \partial^2/\partial y^2 = (\Delta_x - \Delta_x^2/2 + \Delta_x^3/3 - \cdots)^2 + (\Delta_y - \Delta_y^2/2 + \Delta_y^3/3 - \cdots)^2$ with the Gröbner bases of the respective function classes of Table A.2, notice that some of the operators are identical (see Table A.5). This means that the structure of filter bank can be simpler than the lattice of Figure A.8(a).

As the same difference operator is optimal for more than one function class, the number of operators also is limited. When two classes

share the same optimal operator for a given window size, we consider only the most general class for fitting because it yields the smallest maximal approximation error ϵ. The following classes share an operator:

- The function classes G_1, G_2, G_3, G_4, G_5 and G_6 form a subset of functions for which the optimal difference operator for the Laplacian is the zero operator. G_6 is the most general of these functions, and is therefore used as a representative for this set of classes.
- G_8 represents the set of quadratic function classes $\{G_7$,$G_8\}$.
- Quadratic functions as in G_9, G_{10}, G_{11} and G_{12} are represented by G_{12}. For 5x5 templates, the optimal operator generated for these classes is the same as that for class G_{13}. Therefore, we use G_{13} to generate the feature detection templates in the filter bank for this subset of function classes.

Figure A.9 shows the results of applying our scheme to compute the Laplacian for the unsmoothed Lena image. In each pixel, we select one of the optimal operators for the above 5 sets of function classes based on the given selection criterion with adapted operator costs. The result of the operator is stored and used to compute edges as the zero crossings of the Laplacian.

When looking at the results, we see that the zero operator is mostly chosen in the larger image regions with homogeneous gray values. In those regions, the criterion cost function favors the linear function classes, and thus the zero operator. Regions around edges appear to be well approximated by the quadratic and higher order function classes.

When we visually evaluate the edges obtained from zero crossings of the Laplacian in Figure A.9(b), we notice that adaptive Laplacian operators using the criterion $\epsilon|R|$ give accurate results, even for an unsmoothed image. When comparing this results to that obtained with the widely applied discrete Laplacian operator of the form

0	1	0
1	-4	1
0	1	0

(A.11)

in Figure A.9(c). This approach gives no acceptable results without smoothing. When comparing to the edges obtained after Gaussian smoothing with kernels of different width (Figure A.9$(d-e)$), our results remain better. There are less spurious edge pixels detected in

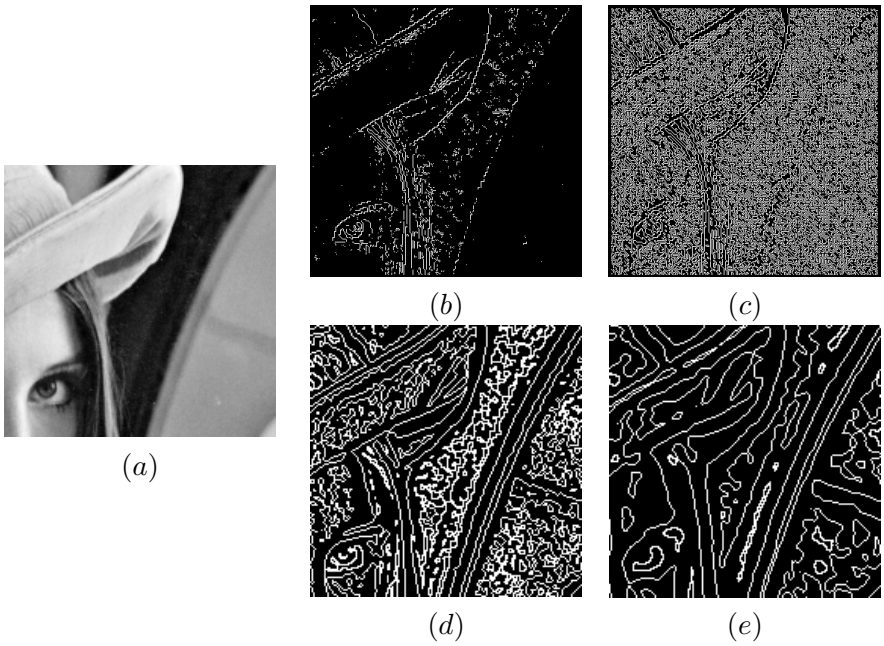


Figure A.9: The original image (a), for which edges are detected as zero crossings after Laplacian computation on the unsmoothed image by the proposed method (b) and by the default method (d). (e) and (f) show the edges obtained by the Laplacian of Gaussian (LoG) method, respectively for images smoothed with a Gaussian of standard deviation $\sqrt{2}$ and $2\sqrt{2}$.

homogeneous image regions, and the edges of finer details are preserved better. Note that some drawbacks of LoG edge detection become more apparent in the results of Figure A.9$(d-e)$ when the images are smoothed to a greater extent. Edges tend to form closed loops, and sharp corners are smoothed too much. The edges of finer (and even coarser) details disappear on higher scales while edges are still detected in (noisy) homogeneous regions.

A.4 Conclusion

We present a mathematical framework to replace a differential operator by a difference operator that is optimal for a specific class of functions. Instead of actually approximating the digitized function by a fitting function of certain class, we apply a filter bank in order to determine locally, for a window D of certain size, how well the digitized

function can be characterized by that class of functions. While this step results in a fitting cost ϵ, for each class and window size, there is also an optimal operator cost $|R|$. For each local part of the function, the best operator is the one that minimizes the combined cost $\epsilon|R|$.

As this cost relies on both the fitting and the operator cost, we can select fitting functions of the most appropriate order in a window of appropriate size, thus striking a balance between over- (or under)fitting, and a sufficiently accurate estimation of the differential operation.

A topic not discussed here is how to employ the freedom left in choosing an operator when there is more than one optimal operator (e.g. when computing derivatives for image functions). This freedom arises when the window is large enough, and allows the use of additional criteria, such as further reduction of noise, or the reduction of the average error, apart from the maximum error of the operator.

As this research topic is work in progress, the methods must still be considered in more detail. Future work will consist not only of improving and extending the current framework. We would like to apply the filters H_i that can be generated for the different function classes toward the description of local image content, and use the responses to the filter bank as a feature or texture description vector for image regions. The choice of which filter subbanks to use for which image region size can then also be adapted to the local image intensity surface.

Original Contributions This research is mainly presented in [Veelaert and Teelen, 2009a]. For a more elaborate discussion, we also refer to [Teelen and Veelaert, 2006], [Veelaert and Teelen, 2008], and [Veelaert and Teelen, 2009b].

Appendix B

Proofs

Proposition 5.2. An affine transformation of a convex combination of points is equal to the convex combination of the transformed points, i.e., $\mathsf{T}(\alpha_1\mathbf{x_1}+\ldots+\alpha_n\mathbf{x_n}) = \alpha_1\mathsf{T}(\mathbf{x_1})+\ldots+\alpha_n\mathsf{T}(\mathbf{x_n})$, with $\alpha_1+\ldots+\alpha_n = 1$ and $0 \leq \alpha_i \leq 1$.

Proof. For two points $\mathbf{x_1} = (x_1, y_1)$ and $\mathbf{x_2} = (x_2, y_2)$, and the convex combination $\mathbf{x}' = \alpha_1\mathbf{x_1} + \alpha_2\mathbf{x_2}$, the affine transformation T of the form (5.5) yields

$$\begin{aligned} x' &= \alpha_1(a_1x_1 + a_2y_1 + a_3) + \alpha_2(a_1x_2 + a_2y_2 + a_3) \\ y' &= \alpha_1(a_4x_1 + a_5y_1 + a_6) + \alpha_2(a_4x_2 + a_5y_2 + a_6). \end{aligned}$$

Because $\alpha_1 + \alpha_2 = 1$, this can be rewritten as

$$\begin{aligned} x' &= a_1(\alpha_1x_1 + \alpha_2x_2) + a_2(\alpha_1y_1 + \alpha_2y_2) + a_3 \\ y' &= a_4(\alpha_1x_1 + \alpha_2x_2) + a_3(\alpha_1y_1 + \alpha_2y_2) + a_6. \end{aligned}$$

It follows that $\alpha_1\mathsf{T}(\mathbf{x_1}) + \alpha_2\mathsf{T}(\mathbf{x_2}) = \mathsf{T}(\alpha_1\mathbf{x_1} + \alpha_2\mathbf{x_2})$. □

Proposition 5.3. The application of a convex combination of transformations to a point $\mathbf{x}$ is equal to the convex combination of that point under the different transformations, i.e., $(\alpha_1\mathsf{T}_1 + \ldots + \alpha_n\mathsf{T}_n)(\mathbf{x}) = \alpha_1\mathsf{T}_1(\mathbf{x}) + \ldots + \alpha_n\mathsf{T}_n(\mathbf{x})$, with $\alpha_1 + \ldots + \alpha_n = 1$ and $0 \leq \alpha_i \leq 1$.

Proof. A point $\mathbf{x} = (x, y)$ is mapped to a point $\mathbf{x}' = (x', y')$ with a convex combination $\alpha_1\mathsf{T}_1(\mathbf{x}) + \alpha_2\mathsf{T}_2(\mathbf{x})$, where T_1 and T_2 are transformations of the form (5.5). By writing down the equations,

$$\begin{cases} x' &= \alpha_1(a_{11}x + a_{12}y + a_{13}) + \alpha_2(a_{21}x + a_{22}y + a_{23}) \\ y' &= \alpha_1(a_{14}x + a_{15}y + a_{16}) + \alpha_2(a_{24}x + a_{25}y + a_{26}) \end{cases}$$

and collecting the terms in x and y, we see that $\mathbf{x}' = (\alpha_1\mathsf{T}_1 + \alpha_2\mathsf{T}_2)(\mathbf{x})$. □

Proposition 5.5. Let $\mathbf{x}$ be a point in $\mathbb{R}^2$, and let $\mathcal{T}$ be a given TUP. Let V_i denote the transformations that correspond to the vertices of the polytope $\mathcal{T}$. Then $R'(\mathbf{x}, \mathcal{T})$ is the convex hull of the points $\mathbf{x}'_\mathbf{i}$, where $\mathbf{x}'_\mathbf{i} = \mathsf{V}_i(\mathbf{x})$.

Proof. By definition, a point $\mathbf{x}'$ lies in $R'(\mathbf{x}, \mathcal{T})$ provided there is a transformation T such that $\mathbf{x}' = \mathsf{T}(\mathbf{x})$, and such that T belongs to $\mathcal{T}$. Since $\mathcal{T}$ is a convex polytope, such a transformation T belongs to the convex span of the vertices V_i of $\mathcal{T}$ (Proposition 5.2). That is, T can be written as $\mathsf{T} = \alpha_1\mathsf{V}_1 + \ldots + \alpha_m\mathsf{V}_m$, with $\alpha_1 + \ldots + \alpha_m = 1$ and $\alpha_1, \ldots, \alpha_m \geq 0$.

It follows that each point $\mathbf{x}'$ in $R'(\mathbf{x}, \mathcal{T})$ can be written as $\mathbf{x}' = \alpha_1\mathsf{V}_1(\mathbf{x}) + \cdots + \alpha_m\mathsf{V}_m(\mathbf{x})$, with $\alpha_1 + \cdots + \alpha_m = 1$ and $\alpha_1, \ldots, \alpha_m \geq 0$ (Proposition 5.3). That is, $\mathbf{x}'$ belongs to the convex span of the points $\mathsf{V}_i(\mathbf{x})$, which proves the proposition. □

Proposition 5.6. Let S be a finite set of points $\mathbf{x_i}$, $\mathcal{R}'$ the collection of the corresponding PURs R'_i. Furthermore, let $\mathcal{T}$ be the TUP determined by the points $\mathbf{x_i}$ and their corresponding regions R'_i, that is $\mathcal{T} = \mathcal{T}(S, \mathcal{R}')$. Then $R'(\mathbf{x_i}, \mathcal{T}) \subseteq R'_i$ for each $\mathbf{x_i} \in S$.

Proof. Since $\mathcal{T}(S, \mathcal{R}') = \cap_i \mathcal{T}(\mathbf{x_i}, R'_i)$, we see that $\mathcal{T}(S, \mathcal{R}') \subseteq \mathcal{T}(\mathbf{x_i}, R'_i)$, $\forall i$. Furthermore, if we set $\mathcal{T}_i = \mathcal{T}(\mathbf{x_i}, R'_i)$, then, by definition, $\mathcal{T}_i$ is the transformation polytope for which $R'(\mathbf{x_i}, \mathcal{T}_i) = R'_i$. The region $R'(\mathbf{x_i}, \mathcal{T})$ will always be part of $R'(\mathbf{x_i}, \mathcal{T}_i)$, but it can be smaller. □

Proposition 5.9. Let $\mathsf{T}_1, \mathsf{T}_2$ be two affine transformations, and let $\mathsf{T} = t\mathsf{T}_1 + (1-t)\mathsf{T}_2$, with $0 \leq t \leq 1$, denote the transformations in their convex span. Let α, β be the parameters of the line $y = \alpha x + \beta$. Then the parameters α', β' of the transformed line lie on a common conic, for $0 \leq t \leq 1$.

Proof. The proof follows by explicit calculation. Although the general case does not represent real difficulties, to limit the size of the expressions, we illustrate the computation here for affine transformations that involve only translations and scaling. Let

$$\mathsf{T}_1 = \begin{bmatrix} a_1 & 0 & e_1 \\ 0 & d_1 & f_1 \\ 0 & 0 & 1 \end{bmatrix}, \mathsf{T}_2 = \begin{bmatrix} a_2 & 0 & e_2 \\ 0 & d_2 & f_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

be two affine transformations, where we assume that $a_1 \neq a_2$, and $d_1 \neq d_2$. Let $(\alpha, -1, \beta)$ be the parameter vector of the line $\alpha x - y + \beta = 0$. Then, the parameters of the line transformed by $\mathsf{T} = t\mathsf{T}_1 + (1-t)\mathsf{T}_2$ are polynomials in t, i.e., $(p', q', r') = (p'(t), q'(t), r'(t))$, for which explicit expressions can be found by using (5.11). The equation $p'(t)x + q'(t)y + r'(t) = 0$, can be rewritten as $y = (-p'(t)/q'(t))x - r'(t)/q(t)$. If we let

$$\begin{aligned} \alpha' &= -p'(t)/q'(t) \\ \beta' &= -r'(t)/q'(t) \end{aligned} \tag{B.2}$$

it follows that the parameter points (α', β') lie on a rational curve parameterized by t.

An implicit equation of the curve can be found by eliminating t from (B.2). The result is a quadratic equation of the form $\tau_{00} + \tau_{10}\alpha' + \tau_{01}\beta' + \tau_{11}\alpha'\beta' + \tau_{20}\alpha'^2 + \tau_{02}\beta'^2 = 0$, with coefficients

$$\begin{aligned} \tau_{00} &= \alpha^2(a_2d_1 - a_1d_2)(-d_2f_1 + d_1f_2) \\ \tau_{01} &= -\alpha^2(d_1 - d_2)(a_2d_1 - a_1d_2) \\ \tau_{10} &= -\alpha(a_2d_1 - a_1d_2)(\beta(a_1d_2 - a_2d_1) + \alpha(d_1e_2 - d_2e_1) + a_2f_1 - a_1f_2) \\ \tau_{11} &= \alpha(a_1 - a_2)(a_2d_1 - a_1d_2) \\ \tau_{20} &= -\alpha(a_2d_1 - a_1d_2)(a_2e_1 - a_1e_2) \\ \tau_{02} &= 0 \end{aligned} \tag{B.3}$$

The case $a_1 = a_2$, $d_1 = d_2$ also leads to a conic, although the parameters are not the same as in B.3. □

Proposition 7.1. A projective transformation of a convex combination of points is equal to the convex combination of the transformed points, i.e., $\mathsf{H}(\alpha_1\mathbf{x_1} + \alpha_2\mathbf{x_2}) = \alpha_1\mathsf{H}(\mathbf{x_1}) + \alpha_2\mathsf{H}(\mathbf{x_2})$, provided $\alpha_1 + \alpha_2 = 1$ with $0 \leq \alpha_i \leq 1$.

Proof. For the convex combination $\mathbf{x}' = \alpha_1\mathbf{x_1} + \alpha_2\mathbf{x_2}$ of two points in homogeneous coordinates $\mathbf{x_1} = (x_1, y_1, w_1)$ and $\mathbf{x_2} = (x_2, y_2, w_2)$, the homography H of the form (7.1) yields

$$\begin{aligned} x' &= \alpha_1(h_1x_1 + h_2y_1 + h_3w_1) + \alpha_2(h_1x_2 + h_2y_2 + h_3w_2) \\ y' &= \alpha_1(h_4x_1 + h_5y_1 + h_6w_1) + \alpha_2(h_4x_2 + h_5y_2 + h_6w_2) \\ w' &= \alpha_1(h_7x_1 + h_8y_1 + h_9w_1) + \alpha_2(h_7x_2 + h_8y_2 + h_9w_2). \end{aligned}$$

Since $\alpha_1 + \alpha_2 = 1$, this can be rewritten as

$$\begin{aligned} x' &= h_1(\alpha_1x_1 + \alpha_2x_2) + h_2(\alpha_1y_1 + \alpha_2y_2) + h_3(\alpha_1w_1 + \alpha_2w_2) \\ y' &= h_4(\alpha_1x_1 + \alpha_2x_2) + h_3(\alpha_1y_1 + \alpha_2y_2) + h_6(\alpha_1w_1 + \alpha_2w_2) \\ w' &= h_7(\alpha_1x_1 + \alpha_2x_2) + h_8(\alpha_1y_1 + \alpha_2y_2) + h_9(\alpha_1w_1 + \alpha_2w_2). \end{aligned}$$

It follows that $\alpha_1 \mathsf{H}(\mathbf{x_1}) + \alpha_2 \mathsf{H}(\mathbf{x_2}) = \mathsf{H}(\alpha_1 \mathbf{x_1} + \alpha_2 \mathbf{x_2})$, and the convex combination of points is preserved by a projective transformation. □

Proposition 7.2. The application of a convex combination of projective transformations to a point $\mathbf{x_i}$ is equal to the convex combination of the transformed points, i.e., $(\alpha_1 \mathsf{H}_1 + \alpha_2 \mathsf{H}_2)(\mathbf{x}) = \alpha_1 \mathsf{H}_1(\mathbf{x}) + \alpha_2 \mathsf{H}_2(\mathbf{x})$, with $\alpha_1 + \alpha_2 = 1$ and $0 \leq \alpha_i \leq 1$.

The proof of the above proposition is straightforward, and can be written along the same lines as for the previous proposition.

Proposition 7.5. Let $\mathsf{H}_1, \mathsf{H}_2$ be two homographies of the form (7.1), and let $\mathsf{H} = t\mathsf{H}_1 + (1-t)\mathsf{H}_2$, with $0 \leq t \leq 1$, denote the transformations in their convex span. Let α, β be the parameters of the line $y = \alpha x + \beta$. Then the parameters α', β' of the line transformed by H lie on a common conic, for $0 \leq t \leq 1$.

The proof can be given by explicit calculation along the same lines as for Proposition 5.9. However, because of the rather large size of the expressions in the general case, we will not consider it in full detail here.

Appendix C

Software

C.1 Feature Extraction and Description

To illustrate the several feature extraction and description methods discussed in chapters 2 and 3, we made use of the software code kindly provided by the authors of [Lowe, 2004] (SIFT), [Bay et al., 2006] (SURF) and [Rosten et al., 2010] (FAST).

C.2 Polytope Representation

The uncertainty of each transformation is described by a polytope $\mathcal{T}$ in a n-dimensional transformation parameter space. Generally, the polytope can be given in two different representations,

- either as a set of inequalities (or halfspaces) of the form (5.6),
- or as a set of vertices of the polytope.

The halfspace representation is more useful to compute the intersection of different polytopes, while the vertices of the polytope are necessary to compute the implied regions.

An efficient conversion from one polytope representation to another can be performed by different algorithms [Fukuda, 2002], of which the software was provided by the authors.

- **Double description**: *cddlib*, *cdd* and *cddr+* by Fukuda [Fukuda, 2002]: *cddlib* is a C-library with basic polyhedral conversion functions and LP solvers. *cddlib* can be compiled with both GMP rational and floating point arithmetic, however, the exact version

cddr+ is much slower. It is efficient for highly degenerate cases, and can remove redundancies from input data using a built-in LP code.

- **Reverse search** algorithm [Avis and Fukuda, 1992], such as in the lrs-package [Avis]: there is a C-implementation available for exact arithmetic only. lrs is efficient for non-degenerate cases and is probably the only available code which can deal with problems generating a large amount of output (e.g. up to a one million vertices or facets).

- **Primal-dual** algorithm [Bremner et al., 1998]: a C-implementation is available for exact arithmetic only. It is efficient for dually non-degenerate cases.

- **Beneath-beyond** method, like in *qhull* [Barber et al., 1996, 2003]: This is the dual of the double description method. *qhull* is available as a C-library for floating arithmetic only, but it handles numerical problems well. *qhull* is highly efficient for non-degenerate cases.

- **Fourier-Motzkin elimination** method [Ziegler, 1995], as implemented in *porta* [Christof]: Efficient C-implementation for combinatorial (e.g. 0-1) polytopes. *porta* guarantees correct numerical results as long as double precision integer arithmetic does not overflow. It can list all integer solutions in a polytope.

- *Polymake* [Gawrilow and Joswig, 2009, 2001]: A computational environment for the algorithmic treatment of convex polytopes and polyhedra. *Polymake* uses cdd+/porta/lrs for representation conversions, and is extendable for new associated structures or data.

Note that the solution for transformation problems in our uncertainty framework requires computations on equations or inequalities in multiple precision arithmetic, otherwise a correct solution is almost never obtained. One can derive approximations in floating point representation. However, it is quite difficult to guarantee the correctness of the computations, when polytopes must be represented by many inequalities, or when intersections must be correctly computed.

Bibliography

W.W. Adams and P. Loustaunau. *An introduction to Gröbner bases*. American Mathematical Society, 1994.

A.P. Ashbrook, N.A. Thacker, P.I. Rockett, and C.I. Brown. Robust recognition of scaled shapes using pairwise geometric histograms. In *Proc. British Machine Vision Conference (BMVC)*, pages 503–512, 1995.

D. Avis. *lrs*. McGill University. Available from http://cgm.cs.mcgill.ca/ avis/C/lrs.html.

D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and computational geometry*, 8(1):295–313, 1992.

D.H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. *qhull*, 2003. Program and report available from http://www.qhull.org/.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Lecture Notes in Computer Science*, 3951:404, 2006.

P.R. Beaudet. Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*, volume 579, page 583, 1978.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

M. Berlamont. Real-time multi-camera bewaking. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2010.

I. Bloch. Fuzzy relative position between objects in image processing: A morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:657–664, 1999a.

I. Bloch. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition*, 32:1873–1895, 1999b.

D. Bremner, K. Fukuda, and A. Marzetta. PrimalDual Methods for Vertex and Facet Enumeration. *Discrete and Computational Geometry*, 20 (3):333–357, 1998.

N. Burrus, T.M. Bernard, and J.M. Jolion. Image segmentation by a contrario simulation. *Pattern Recognition*, 42(7):1520–1532, 2009.

L. Buzer. An elementary algorithm for digital line recognition in the general case. In *Discrete Geometry in Computer Imagery*, pages 299–310. Springer, 2005.

M. Callewaert, J. De Ryck, D. De Clercq, and J. Bourgois. Experimentele studie naar de bewegingsanalytische aspecten van ergometrie van het zeilen in eenmansboten. Master's thesis, Universiteit Gent, Ghent, Belgium, 2009.

J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, pages 679–698, 1986.

T. Christof. *PORTA - A Polyhedron Representation Transformation Algorithm*. Available from the ZIB electronic library ELIB via elib@zib-berlin.de or http://www.zib.de/Optimization/Software/Porta/.

O. Chum. *Two-view geometry estimation by random sample and consensus*. PhD thesis, Czech Technical University, 2005.

O. Chum and J. Matas. Matching with PROSAC-progressive sample consensus. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 220. IEEE Computer Society; 1999, 2005.

O. Chum and J. Matas. Optimal randomized RANSAC. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1472–1482, 2008.

D. Coeurjolly and R. Klette. A comparative evaluation of length estimators. In *International Conference on Pattern Recognition*, volume 16, pages 330–334, 2002.

L. Collatz and P.G. William. *The numerical treatment of differential equations*. Springer Berlin, 1960.

J. Coppens and T. Van Severen. Real-time visiesysteem voor de analyse van de rij-omgeving. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2008.

D.A. Cox, J.B. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 1997.

A. Criminisi. *Accurate Visual Metrology from Single and Multiple Uncalibrated Images*. Springer-Verlag London Ltd., 2001.

G. Csurka, C. Zeller, Z. Zhang, and O.D. Faugeras. Characterizing the uncertainty of the fundamental matrix. *Computer Vision and Image Understanding*, 68(1):18–36, 1997.

T. De Gols and T. Pauwels. General Purpose Computing op GPU voor beeldverwerking. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2009.

I. Debled-Renesson and J-P. Reveilles. A linear algorithm for segmentation of discrete curves. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(6):635–662, 1995.

F. Deboeverie, P. Veelaert, K. Teelen, and W. Philips. Face Recognition Using Parabola Edge Map. *Lecture Notes in Computer Science*, 5259: 994–1005, 2008.

F. Deboeverie, K. Teelen, P. Veelaert, and W. Philips. Vehicle Tracking using Geometric Features. *Lecture Notes in Computer Science*, 5807: 506–515, 2009.

D. Demigny, T. Kamle, and P. Etis-Ensea. A discrete expression of Canny's criteria for step edge detectorperformances evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1199–1211, 1997.

E. Dubrofsky and R.J. Woodham. Combining line and point correspondences for homography estimation. In *Proceedings of the 4th International Symposium on Advances in Visual Computing, Part II*, page 213. Springer, 2008.

R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15:11–15, 1972.

M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

W. Förstner. A framework for low level feature extraction. In *Proceedings of 3rd European Conference on Computer Vision (ECCV'94)*, page 383. Springer-Verlag, 1994.

W. Förstner. Uncertainty and Projective Geometry. *Handbook of Computational Geometry for Pattern Recognition, Computer Vision, Neurocomputing and Robotics, Springer*, 2004.

W. Förstner. Uncertainty and projective geometry. *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*, pages 493–535, 2005.

W. Förstner and E. Gulch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

Franquin. *Guust Flater*. Dupuis & Marsu Productions, 1997.

K. Fukuda. *cdd, cddplus and cddlib homepage*. McGill University, 2002. http://www.cs.mcgill.ca/ fukuda/software/cdd home/cdd.html.

Y. Gao and M.K.H. Leung. Face recognition using line edge map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):767–779, 2002.

E. Gawrilow and M. Joswig. Polymake: an approach to modular software design in computational geometry. In *Proceedings of the 17th annual symposium on Computational geometry*, pages 222–231. ACM New York, NY, USA, 2001.

E. Gawrilow and M. Joswig. *polymake homepage*. TU Berlin / TU Darm-Stadt, 2009. http://www.math.tu-berlin.de/polymake/.

V.P. Gerdt, Y.A. Blinkov, and V.V. Mozzhilkin. Grobner Bases and Generation of Difference Schemes for Partial Differential Equations. *Symmetry, Integrability and Geometry: Methods and Applications (SIGMA)*, 2, 2006.

R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete mathematics, Second Edition*. Addison-Wesley Reading, Massachusetts, 1994.

S.R. Gunn. On the discrete representation of the Laplacian of Gaussian. *Pattern Recognition*, 32(8):1463–1472, 1999.

D. Haeck. Detectie en herkenning van weggebruikers in videosequenties. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2008.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50, 1988.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

E. Helly. Uber Mengen konvexer Korper mit gemeinschaftlichen Punkten. *Jahresber. Deutsch. Math.-Verein*, 32:175–176, 1923.

P. Henrici. *Elements of numerical analysis*. Wiley New York, 1964.

P. V. C. Hough. Method and means for recognizing complex patterns, 1962. US Patent 3,069,654.

F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 2, pages 90–96, 2004.

K. Kanatani. *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science Inc. New York, NY, USA, 1996.

K. Kanatani. Uncertainty Modeling and Model Selection for Geometric Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1307–1319, 2004.

K. Kanatani. *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*, chapter Uncertainty Modeling and Geometric Inference, pages 461–492. Bayro Corrochano, E., Springer, 2005.

Y. Kanazawa and K. Kanatani. Do we really have to consider covariance matrices for image features? In *Proceedings of the 8th IEEE International Conference on Computer Vision(ICCV)*, volume 2, 2001.

Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 2, pages 506–513. IEEE Computer Society, 2004.

A. Kheyrollahi and T.P. Breckon. Automatic real-time road marking recognition using a feature driven approach. *Machine Vision and Applications*, page published online 13 August 2010, 2010.

L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, pages 95–102, 1982.

R. Klette and A. Rosenfeld. *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann, 2004.

K. Knaepkens. Computervisie-analyse van voetbalmatchen. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2009.

J.J. Koenderink and A.J. van Doorn. Representation of local geometry in the visual system. *Biological cybernetics*, 55(6):367–375, 1987.

U. Kothe, P. Stelldinger, and H. Meine. Provably Correct Edgel Linking and Subpixel Boundary Reconstruction. *Lecture Notes in Computer Science*, 4174:81–90, 2006.

J.O. Lachaud, A. Vialard, and F. de Vieilleville. Analysis and comparative evaluation of discrete tangent estimators. *Lecture notes in computer science (DGCI)*, 3429:240–251, 2005.

O. Laligant and F. Truchetet. A Nonlinear Derivative Scheme Applied to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):242–257, 2010.

M. Larcenet. *Le Combat ordinaire*. Dargaud, 2003.

T. Lindeberg. Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. *Journal of Mathematical Imaging and Vision*, 3(4):349–376, 1993.

T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of applied statistics*, 21:224–270, 1994.

H. Ling and D.W. Jacobs. Deformation invariant image matching. In *10th IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1466–1473, 2005.

H.C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, page 61, 1987.

D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

M. Maertens. Automatische herkenning van verkeersinfrastructuur. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2006.

J. Matas and O. Chum. Randomized RANSAC with Td, d test. *Image and Vision Computing*, 22(10):837–842, 2004.

J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. *Lecture Notes in Computer Science*, pages 128–142, 2002.

K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1):43–72, 2005.

F. Mindru, T. Tuytelaars, L. Van Gool, and T. Moons. Moment invariants for recognition under changing viewpoint and illumination. *Computer Vision and Image Understanding*, 94(1-3):3–27, 2004.

H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, page 584. Cambridge, Massachusetts, USA, 1977.

J.M. Morel and G. Yu. ASIFT: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2): 438–469, 2009.

P. Musé, F. Sur, F. Cao, Y. Gousseau, and J.M. Morel. An a contrario decision method for shape element recognition. *International Journal of Computer Vision*, 69(3):295–315, 2006.

J.A. Noble. Finding corners. *Image and Vision Computing*, 6(2):121–128, 1988.

T. Noda, T. Takahashi, D. Deguchi, I. Ide, H. Murase, Y. Kojima, and T. Naito. Recognition of Road Markings from In-Vehicle Camera Images by a Generative Learning Method. *Proc. of 11th IAPR Conference on Machine Vision Applications*, pages 514–517, 2009.

U. Oberst. Multidimensional constant linear systems. *Acta Applicandae Mathematicae*, 20(1-2):1–175, 1990.

U. Oberst and F. Pauer. The Constructive Solution of Linear Systems of Partial Difference and Differential Equations with Constant Coefficients. *Multidimensional Systems and Signal Processing*, 12(3):253–308, 2001.

C. Perwass and W. Förstner. Uncertain Geometry with Circles, Spheres and Conics. *Computational Imaging and Vision - Geometric Properties for Incomplete Data*, 31:23–41, 2006.

C. Perwass, C. Gebken, and G. Sommer. Estimation of Geometric Entities and Operators from Uncertain Data. *Lecture Notes in Computer Science*, 3663:459–467, 2005.

R. Raguram, J.M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. *Proceedings of the 10th European Conference on Computer Vision*, pages 500–513, 2008.

R. Raguram, J.M. Frahm, and M. Pollefeys. Exploiting Uncertainty in Random Sample Consensus. In *Proc. 12th Int. Conf. on Computer Vision (ICCV)*, pages 2074–2081, 2009.

T. Randen and J.H. Husoy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310, 1999.

X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. 9th Int. Conf. Computer Vision*, volume 1, pages 10–17, 2003.

X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1214–1221, 2005.

R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

P.L. Rosin. Techniques for Assessing Polygonal Approximationsof Curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):659–666, 1997.

P.L. Rosin. Assessing the behaviour of polygonal approximation algorithms. *Pattern recognition*, 36(2):505–518, 2003.

E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508–1511, 2005.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, 2006.

E. Rosten, R. Porter, and T. Drummond. Faster and better: a machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2010.

F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (3):477–491, 2007.

P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.

C. Schmid and A. Zisserman. Automatic line matching across views. In *1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997. Proceedings.*, pages 666–671, 1997.

C. Schmid and A. Zisserman. The geometry and matching of lines and curves over multiple views. *International Journal of Computer Vision*, 40(3):199–233, 2000b.

C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

S. Se, D. Lowe, and J. Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21(8):735, 2002.

S. Se, H.K. Ng, P. Jasiobedzki, and T.J. Moyung. Vision based modeling and localization for planetary exploration rovers. In *Proceedings of International Astronautical Congress*, 2004.

J. Shi and C. Tomasi. Good features to track. In *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94.*, pages 593–600, 1994.

S.N. Sinha, J.M. Frahm, M. Pollefeys, and Y. Genc. GPU-based video feature tracking and matching. In *EDGE, Workshop on Edge Computing Using New Commodity Architectures*, volume 278. Citeseer, 2006.

S.M. Smith and J.M. Brady. SUSAN-A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

J. Stoer and C. Witzgall. *Convexity and Optimization in Finite Dimensions*. Springer-Verlag, 1970.

F. Sur, N. Noury, and M-O. Berger. Computing the uncertainty of the 8 point algorithm for fundamental matrix estimation. In *Proceedings of the 19th British Machine Vision Conference (BMVC)*, United Kingdom, 2008.

K. Teelen and P. Veelaert. Computing the Uncertainty of Geometric Primitives and Transformations. In *Proceedings of ProRISC*, pages 317–325, 2004a.

K. Teelen and P. Veelaert. Uncertainty of affine transformations in digital images. *Proceedings of ACIVS 2004*, pages 23–30, 2004b.

K. Teelen and P. Veelaert. Confidence measures for consensus sets in transformation uncertainty. In *Proceedings of ProRISC 2005*, pages 679–686, 2005a.

K. Teelen and P. Veelaert. Computing the uncertainty of transformations in digital images. In *IS&T/SPIE Electronic Imaging, Vision Geometry XIII*, pages 1–15, 2005b.

K. Teelen and P. Veelaert. Image Registration Using Uncertainty Transformations. *Lecture Notes in Computer Science*, 3708:348–355, 2005c.

K. Teelen and P. Veelaert. Improving Difference Operators by Local Feature Detection. *Lecture Notes in Computer Science*, 4245:391–402, 2006.

K. Teelen and P. Veelaert. Transformation Polytopes for Line Correspondences in Digital Images. *Lecture Notes in Computer Science*, 4958: 238–249, 2008.

K. Teelen and P. Veelaert. Computing regions of interest for geometric features in digital images. *Discrete Applied Mathematics*, 157(16):3457 – 3472, 2009.

L. Tessens. *Information Selection and Fusion in Vision Systems*. PhD thesis, Ghent University, 2010.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

B. Tordoff and R. Cipolla. Uncertain RanSaC. *Proc. IAPR Workshop on Machine Vision Applications*, pages 594–597, 2005.

B. Tordoff and D. Murray. Guided sampling and consensus for motion estimation. *Proceedings of the 7th European Conference on Computer Vision*, pages 82–96, 2002.

P.H.S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):35–61, 2002.

P.H.S. Torr and C. Davidson. IMPSAC: Synthesis of importance sampling and random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):354–364, 2003.

P.H.S. Torr and A. Zisserman. Robust parameterization and computation of the trifocal tensor. *Image and Vision Computing*, 15(8):591–605, 1997.

P.H.S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

M. Trajković and M. Hedley. Fast corner detection. *Image and Vision Computing*, 16(2):75–87, 1998.

T. Tuytelaars and K. Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3): 177–280, 2008.

T. Tuytelaars and L. Van Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 412–425, 2000.

T. Tuytelaars and L. Van Gool. Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59 (1):61–85, 2004.

G. Valiente. *Algorithms on trees and graphs*. Springer Verlag, 2002.

D. Van Hamme, P. Veelaert, W. Philips, K. Teelen, N. Stevens, and Vermeersch B. Foliage Recognition Based on Local Edge Information. *Lecture Notes in Computer Science*, 5259:838–849, 2008a.

D. Van Hamme, P. Veelaert, W. Philips, K. Teelen, N. Stevens, and B. Vermeersch. Texture-based foliage detection using edge proximity. In *Proceedings of ProRISC 2008*, pages 322–327, 2008b.

M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 691–698, 2003.

P. Veelaert. On the flatness of digital hyperplanes. *Journal of Mathematical Imaging and Vision*, 3(2):205–221, 1993.

P. Veelaert. Local feature detection for digital surfaces. In *Proc. of the SPIE conference on Vision Geometry V*, volume 2826, pages 34–45. SPIE, 1996.

P. Veelaert. Algorithms that measure parallelism and concurrency of lines in digital images. In *Proc. of the SPIE Conference on Vision Geometry VIII*, volume 3811, pages 69–79, 1999a.

P. Veelaert. Geometric constructions in the digital plane. *Journal of Mathematical Imaging and Vision*, 11(2):99–118, 1999b.

P. Veelaert. Parallel Line Grouping Based on Interval Graphs. *Lecture Notes in Computer Science*, 1953:530–542, 2000.

P. Veelaert. Collinearity and Weak Collinearity in the Digital Plane. *Lecture Notes in Computer Science*, 2243:439–454, 2001.

P. Veelaert. Concurrency of Line Segments in Uncertain Geometry. *Lecture Notes in Computer Science*, 2301:289–300, 2002.

P. Veelaert. Graph-theoretical properties of parallelism in the digital plane. *Discrete Applied Mathematics*, 125(1):135–160, 2003a.

P. Veelaert. Reestablishing consistency of uncertain geometric relations in digital images. *Lecture notes in computer science*, 2616:268–281, 2003b.

P. Veelaert. Uncertain geometry in computer vision. *Lecture Notes in Computer Science*, 3429:359–370, 2005.

P. Veelaert and K. Teelen. Consensus sets for affine transformation uncertainty polytopes. *Computers & Graphics*, 30(1):77–85, 2006a.

P. Veelaert and K. Teelen. Fast Polynomial Segmentation of Digitized Curves. *Lecture Notes in Computer Science*, 4245:482–493, 2006b.

P. Veelaert and K. Teelen. Optimal Difference Operator Selection. *Lecture Notes in Computer Science*, 4992:495–506, 2008.

P. Veelaert and K. Teelen. Adaptive and optimal difference operators in image processing. *Pattern Recognition*, 42:2317–2326, 2009a.

P. Veelaert and K. Teelen. Feature controlled adaptive difference operators. *Discrete Applied Mathematics*, 157:571–582, 2009b.

T. Veit, F. Cao, and P. Bouthemy. An a contrario decision framework for region-based motion detection. *International Journal of Computer Vision*, 68(2):163–178, 2006.

J. Verschuere. Visuele evaluatie van beweging. Master's thesis, Hogeschool Gent, Ghent, Belgium, 2010.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, volume 1, pages 511–518, 2001.

Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):161–195, 1998.

Z. Zhang, R. Deriche, O. Faugeras, and Q.T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial Intelligence*, 78(1-2):87–119, 1995.

G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics 152. Springer-Verlag, New York, 1995.

B. Zitová and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977–1000, 2003.

M. Zuliani, C. Kenney, and BS Manjunath. A mathematical comparison of point detectors. In *Conference on Computer Vision and Pattern Recognition*, volume 11, pages 172–178, 2004.